LSF Reference Guide

Version 4.1
December 2000

Platform Computing Corporation

We'd Like to Hear from You

You can help us make this manual better by telling us what you think of the content, organization, and usefulness of the information. If you find an error, or just want to make a suggestion for improving this manual, please address your comments to doc@platform.com.

Your comments should pertain only to the LSF documentation. For product support, contact support@platform.com.

Revision Information

This document has been revised as follows:

Edition	Description
First	For LSF Version 4.0
Second	Revised for LSF Version 4.1

Contents

Part III: Configuration Files

Part IV: Troubleshooting

Welcome

Overview This chapter describes how to get the most out of this guide, how to find more information about LSF, and how to get technical assistance using LSF.

Contents

About This Guide

Purpose of This Guide

This guide provides reference information on the following LSF topics:

- LSF commands
- Environment variables
- Configuration files
- Troubleshooting

Who Should Use This Guide

This guide accompanies the *LSF Administrator's Guide*, and is your source for reference information.

Typographical Conventions

Typeface	Meaning	Example
`Courier`	The names of on-screen computer output, commands, files, and directories.	The `lsid` command
`Bold Courier`	What you type must be exactly as shown.	Type **`cd /bin`**
Italics	◆ Book titles, new words or terms, or words to be emphasized. ◆ Command-line place holders—replace with a real name or value.	The queue specified by *queue_name*

Command Notation

Notation	Meaning	Example
Quotes " or '	Must be entered exactly as shown.	"*job_ID*[*index_list*]"
Commas ,	Must be entered exactly as shown.	-C *time0*,*time1*
Ellipsis …	The agument before the ellipsis can be repeated. Do not enter the ellipsis.	*job_ID* ...
lower case italics	The argument must be replaced with a real value you provide.	*job_ID*
OR bar |	You must enter one of the items separated by the bar. You cannot enter more than one item. Do not enter the bar.	[-h | -V]
Parenthesis ( )	Must be entered exactly as shown.	-X "*exception_cond*([*params*]) ::*action*] ...
Option or variable in square brackets [ ]	The argument within the brackets is optional. Do not enter the brackets.	lsid [-h]
Shell prompts	◆ C shell: % ◆ Bourne shell and Korn shell: $ ◆ root account: # Unless otherwise noted, the C shell prompt is used in all command examples.	`% cd /bin`

Learning About LSF

Finding LSF Information

Information about LSF is available from the following sources:

- World Wide Web and FTP
- README files and Release Notes
- Printed LSF manuals
- Online documentation
- LSF Technical support

World Wide Web and FTP

The latest information about all supported releases of LSF is available on the Platform Computing Corporation site on the World Wide Web at `http://www.platform.com`. Look in the Online Support area for current README files, Release Notes, Upgrade Notices, Frequently Asked Questions (FAQs), Troubleshooting, and other helpful information.

The Platform FTP site (`ftp.platform.com`) also provides current README files and Release Notes for all supported releases of LSF.

If you have problems accessing the Platform web site or the Platform FTP site, send email to `info@platform.com`.

README Files and Release Notes

The LSF distribution files contain the current README files and Release Notes. Before installing LSF, be sure to read the files named `readme.html` and `release_notes.html`. They contain important installation and configuration information that is not in the printed documentation. You can also view these files from the Download area of the LSF Online Support Web page.

LSF Manuals

The following LSF manuals are available:

- *LSF UNIX Installation Guide*
- *LSF Administrator's Guide*
- *LSF Reference Guide*
- *LSF JobScheduler Administrator's Guide*
- *LSF JobScheduler User's Guide*
- *LSF Analyzer Guide*
- *LSF Analyzer WebAccess*
- *LSF Analyzer Oracle Plug-in Guide*
- *LSF Parallel User's Guide*
- *LSF Programmer's Guide*

Online Documentation

The following information is available online:

- LSF manuals in HTML and PDF format, available on the LSF product CD, and the Platform web site
- Man pages (accessed with the `man` command) for all LSF commands
- Online help available through the Help menu for the LSF graphical applications: `xlsbatch`, `xbmod`, `xbsub`, `xlsmon`, and `xlsadmin`

Technical Support

Contact Platform Computing or your LSF vendor for technical support.

Email support@platform.com

World Wide Web http://www.platform.com

Phone
- North America: +1 905 948 4297
- Europe: +44 1256 370 530
- Asia: +86 1062 381125

Toll-free Phone 1-877-444-4LSF (+1 877 444 4573)

Mail
LSF Technical Support
Platform Computing Corporation
3760 14th Avenue
Markham, Ontario
Canada L3R 3T7

When contacting Platform, please include the full name of your company.

We'd Like to Hear from You

If you find an error in this guide, or any LSF manual, or you have a suggestion for improving it, please let us know:

Email doc@platform.com

Mail
LSF Information Development
Platform Computing Corporation
3760 14th Avenue
Markham, Ontario
Canada L3R 3T7

Be sure to tell us:

- The title of the manual you are commenting on
- The version of the product you are using
- The format of the manual (HTML or PDF)

PART

I

Commands

bacct

displays accounting statistics about finished jobs

SYNOPSIS

bacct [**-b** | **-l**] [**-d**] [**-e**] [**-w**] [**-C** *time0*,*time1*] [**-D** *time0*,*time1*] [**-f** *logfile_name*] [**-m** *host_name ...*] [**-N** *host_name* | **-N** *host_model* | **-N** *CPU_factor*] [**-P** *project_name ...*] [**-q** *queue_name ...*] [**-S** *time0*,*time1*] [**-u** *user_name ...* | **-u** `all`] [*job_ID ...*]

bacct [**-h** | **-V**]

DESCRIPTION

Displays accounting statistics about finished jobs.

By default, displays accounting statistics for all finished jobs (with a DONE or EXIT status) submitted by the user who invoked the command, on all hosts, projects, and queues in the LSF system.

By default, `bacct` displays statistics for all jobs logged in the current LSF accounting log file: `LSB_SHAREDIR/`*cluster_name*`/logdir/lsb.acct` (see `lsb.acct(5)`).

By default, CPU time is not normalized.

If neither `-l` nor `-b` is present, displays the fields in SUMMARY only (see OUTPUT).

Statistics not reported by `bacct` but of interest to individual system administrators can be generated by directly using `awk(1)` or `perl(1)` to process the `lsb.acct` file.

All times are in seconds.

Throughput Calculation

The throughput (T) of the LSF system, certain hosts, or certain queues is calculated by the formula:

```
T = N/(ET-BT)
```

where:

- N is the total number of jobs for which accounting statistics are reported

- BT is the Start time—when the first job was logged

- ET is the End time—when the last job was logged

You can use the option `-C` *time0*`,`*time1* to specify the Start time as *time0* and the End time as *time1*. In this way, you can examine throughput during a specific time period.

Jobs involved in the throughput calculation are only those being logged (that is, with a DONE or EXIT status). Jobs that are running, suspended, or that have never been dispatched after submission are not considered, because they are still in the LSF system and not logged in `lsb.acct`.

The total throughput of the LSF system can be calculated by specifying `-u all` without any of the `-m`, `-q`, `-S`, `-D` or *job_ID* options. The throughput of certain hosts can be calculated by specifying `-u all` without the `-q`, `-S`, `-D` or *job_ID* options. The throughput of certain queues can be calculated by specifying `-u all` without the `-m`, `-S`, `-D` or *job_ID* options.

OPTIONS

`-b`

Brief format. Displays accounting statistics in brief format. See OUTPUT for a description of information that is displayed.

`-d`

Displays accounting statistics for only successfully completed jobs (with a DONE status).

`-e`

Displays accounting statistics for only exited jobs (with an EXIT status).

`-l`

Long format. Displays additional accounting statistics. See OUTPUT for a description of information that is displayed.

`-w`

Wide format. Displays accounting statistics in a wide format. No truncation is performed.

-C *time0,time1*

Displays accounting statistics for only jobs that completed or exited during the specified time interval. Reads `lsb.acct` and all archived log files (`lsb.acct.`*n*) unless `-f` is also used.

The time format is the same as in `bhist(1)`.

-D *time0,time1*

Displays accounting statistics for only jobs dispatched during the specified time interval. Reads `lsb.acct` and all archived log files (`lsb.acct.`*n*) unless `-f` is also used.

The time format is the same as in `bhist(1)`.

-f *logfile_name*

Searches only the specified job log file for accounting statistics. Specify either an absolute or relative path.

Useful for offline analysis.

-m *host_name ...*

Displays accounting statistics for only jobs dispatched to the specified hosts.

If a list of hosts is specified, host names must be separated by spaces and enclosed in quotation marks (") or (').

-N *host_name* **| -N** *host_model* **| -N** *CPU_factor*

Normalizes CPU time by the CPU factor of the specified host or host model, or by the specified CPU factor.

If you use `bacct` offline by indicating a job log file, you must specify a CPU factor.

Use `lsinfo` to get host model and CPU factor information.

-P *project_name ...*

Displays accounting statistics for only jobs belonging to the specified projects. If a list of projects is specified, project names must be separated by spaces and enclosed in quotation marks (") or (').

-q *queue_name ...*

Displays accounting statistics for only jobs submitted to the specified queues.

If a list of queues is specified, queue names must be separated by spaces and enclosed in quotation marks (") or (').

-S *time0,time1*

Displays accounting statistics for only jobs submitted during the specified time interval. Reads `lsb.acct` and all archived log files (`lsb.acct.`*n*) unless `-f` is also used.

The time format is the same as in `bhist(1)`.

-u *user_name ...*|**-u all**

Displays accounting statistics for only jobs submitted by the specified users, or by all users if the keyword `all` is specified.

If a list of users is specified, user names must be separated by spaces and enclosed in quotation marks (") or ('). You can specify both user names and user IDs in the list of users.

job_ID ...

Displays accounting statistics for only jobs with the specified job IDs.

This option overrides all other options except `-b`, `-l`, `-f`, `-h`, and `-V`. If the reserved job ID 0 is used, it will be ignored.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

OUTPUT

SUMMARY (default format)

Statistics on jobs. The following fields are displayed:

- Total number of done jobs
- Total number of exited jobs
- Total CPU time consumed
- Average CPU time consumed
- Maximum CPU time of a job
- Minimum CPU time of a job
- Total wait time in queue
- Average wait time in queue

- Maximum wait time in queue
- Minimum wait time in queue
- Average turnaround time
- Maximum turnaround time
- Minimum turnaround time
- Average hog factor of a job
- Maximum hog factor of a job
- Minimum hog factor of a job
- Total throughput
- Beginning time: the completion or exit time of the first job selected
- Ending time: the completion or exit time of the last job selected

The total, average, minimum, and maximum statistics are on all specified jobs.

The wait time is the elapsed time from job submission to job dispatch.

The turnaround time is the elapsed time from job submission to job completion.

The hog factor is the amount of CPU time consumed by a job divided by its turnaround time.

The throughput is the number of completed jobs divided by the time period to finish these jobs. For more details, see DESCRIPTION.

Brief Format (-b)

In addition to the default format SUMMARY, displays the following fields:

U/UID

Name of the user who submitted the job. If LSF fails to get the user name by `getpwuid(3)`, the user ID is displayed.

QUEUE

Queue to which the job was submitted.

SUBMIT_TIME

Time when the job was submitted.

CPU_T

CPU time consumed by the job.

WAIT

Wait time of the job.

TURN

Turnaround time of the job.

FROM

Host from which the job was submitted.

EXEC_ON

Host or hosts to which the job was dispatched to run.

JOB_NAME

Name of the job (see `bsub(1)`).

Long Format (-l)

In addition to the fields displayed by default in SUMMARY and by `-b`, displays the following fields:

JOBID

Identifier that LSF assigned to the job.

PROJECT_NAME

Project name assigned to the job.

STATUS

Status that indicates the job was either successfully completed (DONE) or exited (EXIT).

DISPAT_TIME

Time when the job was dispatched to run on the execution hosts.

COMPL_TIME

Time when the job exited or completed.

HOG_F

Hog factor, equal to "CPU time" / "turnaround time".

MEM

Maximum resident memory usage of all processes in a job, in kilobytes.

SWAP

Maximum virtual memory usage of all processes in a job, in kilobytes.

CWD

Current working directory of the job.

INPUT_FILE

File from which the job reads its standard input (see `bsub`(1)).

OUTPUT_FILE

File to which the job writes its standard output (see `bsub`(1)).

ERR_FILE

File in which the job stores its standard error output (see `bsub`(1)).

FILES

Reads `lsb.acct`, `lsb.acct.`*n*.

SEE ALSO

`bhist(1), bsub(1), bjobs(1), lsb.acct(5)`

badmin

administrative tool for LSF

SYNOPSIS

badmin *subcommand*

badmin [**-h** | **-V**]

SUBCOMMAND LIST

ckconfig [**-v**]
reconfig [**-v**] [**-f**]
mbdrestart [**-v**] [**-f**]
qopen [*queue_name* ... | **all**]
qclose [*queue_name* ... | **all**]
qact [*queue_name* ... | **all**]
qinact [*queue_name* ... | **all**]
qhist [**-t** *time0,time1*] [**-f** *logfile_name*] [*queue_name* ...]
hopen [*host_name* ... | *host_group* ... | **all**]
hclose [*host_name* ... | *host_group* ... | **all**]
hrestart [**-f**] [*host_name* ...| **all**]
hshutdown [**-f**] [*host_name* ... | **all**]
hstartup [**-f**] [*host_name* ... | **all**]
hhist [**-t** *time0,time1*] [**-f** *logfile_name*] [*host_name* ...]
mbdhist [**-t** *time0,time1*] [**-f** *logfile_name*]
hist [**-t** *time0,time1*] [**-f** *logfile_name*]
help [*command* ...] | **?** [*command* ...]
quit
sbddebug [**-c** *class_name* ...] [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**] [*host_name* ...]
mbddebug [**-c** *class_name* ...] [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**]
sbdtime [**-l** *timing_level*] [**-f** *logfile_name*] [**-o**] [*host_name* ...]
mbdtime [**-l** *timing_level*] [**-f** *logfile_name]* [**-o**]

DESCRIPTION

This command can only be used by LSF administrators.

`badmin` provides a set of commands to control and monitor LSF. If no subcommands are supplied for `badmin`, `badmin` prompts for a command from standard input. Commands `bqc`(8), `breconfig`(8) and `breboot`(8) are superceded by `badmin`(8).

Information about each command is available through the `help` command.

The `badmin` commands consist of a set of privileged commands and a set of non-privileged commands. Privileged commands can only be invoked by root or LSF administrators as defined in the configuration file (see `lsf.cluster.cluster`(5) for `ClusterAdmin`). Privileged commands are:

```
reconfig
mbdrestart
qopen
qclose
qact
qinact
hopen
hclose
hrestart
hshutdown
hstartup
```

The configuration file `lsf.sudoers`(5) has to be set in order to use the privileged command `hstartup` by a non-root user.

All other commands are non-privileged commands and can be invoked by any LSF user. If the privileged commands are to be executed by an LSF administrator, `badmin` must be installed `setuid root`, because it needs to send the request using a privileged port.

For subcommands for which multiple host names or host groups can be specified, do not enclose the multiple names in quotation marks.

OPTIONS

subcommand

Executes the specified subcommand. See Usage section.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

USAGE

ckconfig [-v]

Checks LSF configuration files. Configuration files are located in the `LSB_CONFDIR/`*cluster_name*`/configdir` directory.

The LSB_CONFDIR variable is defined in `lsf.conf` (see `lsf.conf`(5)) which is in LSF_ENVDIR or `/etc` (if LSF_ENVDIR is not defined).

By default, `badmin ckconfig` displays only the result of the configuration file check. If warning errors are found, `badmin` prompts you to display detailed messages.

-v

Verbose mode. Displays detailed messages about configuration file checking to stderr.

reconfig [-v] [-f]

Dynamically reconfigures LSF without restarting MBD. Configuration files are checked for errors and the results displayed to stderr. If no errors are found in the configuration files, a reconfiguration request is sent to MBD and configuration files are reloaded.

With this command, MBD is not restarted and `lsb.events` is not replayed. To restart MBD and replay `lsb.events`, use `badmin mbdrestart`.

When you issue this command, MBD is available to service requests while reconfiguration files are reloaded. Configuration changes made since system boot or the last reconfiguration take effect.

If warning errors are found, `badmin` prompts you to display detailed messages. If fatal errors are found, reconfiguration is not performed, and `badmin` exits.

If you add a host to a host group, or a host to a queue, the new host will not be recognized by jobs that were submitted before you reconfigured. If you want the new host to be recognized, you must use the command `badmin mbdrestart`.

`-v`

Verbose mode. Displays detailed messages about the status of the configuration files. Without this option, the default is to display the results of configuration file checking. All messages from the configuration file check are printed to stderr.

`-f`

Disables interaction and proceeds with reconfiguration if configuration files contain no fatal errors.

`mbdrestart [-v] [-f]`

Dynamically reconfigures LSF and restarts MBD. Configuration files are checked for errors and the results printed to `stderr`. If no errors are found, configuration files are reloaded, MBD is restarted, and events in `lsb.events` are replayed to recover the running state of the last MBD. MBD is unavailable to service requests while it restarts.

If warning errors are found, `badmin` prompts you to display detailed messages. If fatal errors are found, MBD restart is not performed, and `badmin` exits.

If `lsb.events` is large, or many jobs are running, restarting MBD can take several minutes. If you only need to reload the configuration files, use `badmin reconfig`.

`-v`

Verbose mode. Displays detailed messages about the status of configuration files. All messages from configuration checking are printed to stderr.

`-f`

Disables interaction and forces reconfiguration and MBD restart to proceed if configuration files contain no fatal errors.

qopen [*queue_name ...* | **all**]

Opens specified queues, or all queues if the reserved word **all** is specified. If no queue is specified, the system default queue is assumed (see **lsb.queues**(5) for DEFAULT_QUEUE). A queue can accept batch jobs only if it is open.

qclose [*queue_name ...* | **all**]

Closes specified queues, or all queues if the reserved word **all** is specified. If no queue is specified, the system default queue is assumed. A queue will not accept any job if it is closed.

qact [*queue_name ...* | **all**]

Activates specified queues, or all queues if the reserved word **all** is specified. If no queue is specified, the system default queue is assumed. Jobs in a queue can be dispatched if the queue is activated.

A queue inactivated by its run windows cannot be reactivated by this command (see **lsb.queues(5)** for RUN_WINDOW).

qinact [*queue_name ...* | **all**]

Inactivates specified queues, or all queues if the reserved word **all** is specified. If no queue is specified, the system default queue is assumed. No job in a queue can be dispatched if the queue is inactivated.

qhist [**-t** *time0,time1*] [**-f** *logfile_name*] [*queue_name ...*]

Displays historical events for specified queues, or for all queues if no queue is specified. Queue events are queue opening, closing, activating and inactivating.

-t *time0,time1*

Displays only those events that occurred during the period from *time0* to *time1*. See **bhist(1)** for the time format. The default is to display all queue events in the event log file (see below).

-f *logfile_name*

Specify the file name of the event log file. Either an absolute or a relative path name may be specified. The default is to use the event log file currently used by the LSF system: `LSB_SHAREDIR/cluster_name/logdir/lsb.events`. Option `-f` is useful for offline analysis.

hopen [*host_name ...* | *host_group ...* | **all**]

Opens batch server hosts. Specify the names of any server hosts or host groups (see `bmgroup(1)`). All batch server hosts will be opened if the reserved word `all` is specified. If no host or host group is specified, the local host is assumed. A host accepts batch jobs if it is open.

hclose [*host_name ...* | *host_group ...* | **all**]

Closes batch server hosts. Specify the names of any server hosts or host groups (see `bmgroup(1)`). All batch server hosts will be closed if the reserved word `all` is specified. If no argument is specified, the local host is assumed. A closed host will not accept any new job, but jobs already dispatched to the host will not be affected. Note that this is different from a host closed by a window - all jobs on it are suspended in that case.

hrestart [**-f**] [*host_name ...* | **all**]

Restarts SBD on the specified hosts, or on all server hosts if the reserved word `all` is specified. If no host is specified, the local host is assumed. SBD will re-execute itself from the beginning. This allows new SBD binaries to be used.

-f

Disables interaction and does not ask for confirmation for restarting SBDs.

hshutdown [**-f**] [*host_name ...* | **all**]

Shuts down SBD on the specified hosts, or on all batch server hosts if the reserved word `all` is specified. If no host is specified, the local host is assumed. SBD will exit upon receiving the request.

-f

Disables interaction and does not ask for confirmation for shutting down SBDs.

hstartup [**-f**] [*host_name ...* | **all**]

Starts up SBD on the specified hosts, or on all batch server hosts if the reserved word `all` is specified. Only `root` and users listed in the file `lsf.sudoers(5)` can use this option, and those users must be able to use `rsh` on all LSF hosts. If no host is specified, the local host is assumed.

-f

Disables interaction and does not ask for confirmation for starting up SBDs.

hhist [**-t** *time0,time1*] [**-f** *logfile_name*] [*host_name ...*]

Displays historical events for specified hosts, or for all hosts if no host is specified. Host events are host opening and closing. Options `-t` and `-f` are exactly the same as those of `qhist` (see above).

mbdhist [**-t** *time0,time1*] [**-f** *logfile_name*]

Displays historical events for MBD. Events describe the starting and exiting of MBD. Options `-t` and `-f` are exactly the same as those of `qhist` (see above).

hist [**-t** *time0,time1*] [**-f** *logfile_name*]

Displays historical events for all the queues, hosts and MBD. Options `-t` and `-f` are exactly the same as those of `qhist` (see above).

help [*command ...*] | **?** [*command ...*]

Displays the syntax and functionality of the specified commands.

quit

Exits the `badmin` session.

sbddebug [**-c** *class_name ...*] [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**] [*host_name ...*]

Sets the message log level for SBD to include additional information in log files. You must be `root` or the LSF administrator to use this command.

If the command is used without any options, the following default values are used:

class_name = 0 (no additional classes are logged)

debug_level = 0 (LOG_DEBUG level in parameter LSF_LOG_MASK)

logfile_name = current LSF system log file in the directory specified by LSF_LOGDIR in the format *daemon_name*`.log.`*host_name*

host_name = local host (host from which command was submitted)

-c *class_name ...*

Specifies software classes for which debug messages are to be logged.

Format of *class_name* is the name of a class, or a list of class names separated by spaces and enclosed in quotation marks.

Possible classes:

LC_AFS - Log AFS messages

LC_AUTH - Log authentication messages

LC_CHKPNT - Log checkpointing messages

LC_COMM - Log communication messages

LC_DCE - Log messages pertaining to DCE support

LC_EEVENTD - Log eeventd messages

LC_EXEC - Log significant steps for job execution

LC_FAIR - Log fairshare policy messages

LC_FILE - Log file transfer messages

LC_HANG - Mark where a program might hang

LC_JGRP - Log job group messages

LC_JLIMIT - Log job slot limit messages

LC_LICENCE - Log license management messages

LC_LOADINDX - Log load index messages

LC_M_LOG - Log multievent logging messages

LC_MPI - Log MPI messages

LC_MULTI - Log messages pertaining to MultiCluster

LC_PEND - Log messages related to job pending reasons

LC_PERFM - Log performance messages

LC_PIM - Log PIM messages

LC_PREEMPT - Log preemption policy messages

LC_SCHED - Log JobScheduler messages

LC_SIGNAL - Log messages pertaining to signals

LC_SYS - Log system call messages

LC_TRACE - Log significant program walk steps

LC_XDR - Log everything transferred by XDR

Note: Classes are also listed in `lsf.h`.

Default: 0 (no additional classes are logged)

-l *debug_level*

Specifies level of detail in debug messages. The higher the number, the more detail that is logged. Higher levels include all lower levels.

Possible values:

0 LOG_DEBUG level in parameter LSF_LOG_MASK in `lsf.conf`.

1 LOG_DEBUG1 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 LOG_DEBUG1, and LOG_DEBUG levels.

2 LOG_DEBUG2 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 LOG_DEBUG1, and LOG_DEBUG levels.

3 LOG_DEBUG3 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2, LOG_DEBUG1, and LOG_DEBUG levels.

Default: 0 (LOG_DEBUG level in parameter LSF_LOG_MASK)

-f *logfile_name*

Specify the name of the file into which debugging messages are to be logged. A file name with or without a full path may be specified.

If a file name without a path is specified, the file will be saved in the directory indicated by the parameter LSF_LOGDIR in `lsf.conf`.

The name of the file that will be created will have the following format:

logfile_name.daemon_name.`log`*.host_name*

If the specified path is invalid, on UNIX, the log file is created in the `/tmp` directory. On Windows NT, no log file is created.

If LSF_LOGDIR is not defined, daemons log to the syslog facility.

Default: current LSF system log file in the directory specified by LSF_LOGDIR in the format *daemon_name.*`log`*.host_name*.

`-o`

Turns off temporary debug settings and resets them to the daemon starting state. The message log level is reset back to the value of LSF_LOG_MASK and classes are reset to the value of LSB_DEBUG_MBD, LSB_DEBUG_SBD.

The log file is also reset back to the default log file.

`host_name ...`

Optional. Sets debug settings on the specified host or hosts.

Lists of host names must be separated by spaces and enclosed in quotation marks.

Default: local host (host from which command was submitted)

`mbddebug` [**`-c`** *`class_name ...`*] [**`-l`** *`debug_level`*] [**`-f`** *`logfile_name`*] [**`-o`**]

Sets message log level for MBD to include additional information in log files. You must be `root` or the LSF administrator to use this command.

See `sbddebug` for an explanation of options.

`sbdtime` [**`-l`** *`timing_level`*] [**`-f`** *`logfile_name`*] [**`-o`**] [*`host_name ...`*]

Sets the timing level for SBD to include additional timing information in log files. You must be `root` or the LSF administrator to use this command.

If the command is used without any options, the following default values are used:

timing_level = no timing information is recorded

logfile_name = current LSF system log file in the directory specified by LSF_LOGDIR in the format *daemon_name*.`log`.*host_name*

host_name = local host (host from which command was submitted)

`-l` *`timing_level`*

Specifies detail of timing information that is included in log files. Timing messages indicate the execution time of functions in the software and are logged in milliseconds.

Valid values: 1 | 2 | 3 | 4 | 5

The higher the number, the more functions in the software that are timed and whose execution time is logged. The lower numbers include more common software functions. Higher levels include all lower levels.

Default: undefined (no timing information is logged)

`-f` *`logfile_name`*

Specify the name of the file into which timing messages are to be logged. A file name with or without a full path may be specified.

If a file name without a path is specified, the file will be saved in the directory indicated by the parameter LSF_LOGDIR in `lsf.conf`.

The name of the file that will be created will have the following format:

logfile_name.daemon_name.`log`.*host_name*

If the specified path is invalid, on UNIX, the log file is created in the `/tmp` directory. On Windows NT, no log file is created.

If LSF_LOGDIR is not defined, daemons log to the syslog facility.

Note: Both timing and debug messages are logged in the same files.

Default: current LSF system log file in the directory specified by LSF_LOGDIR in the format *daemon_name*.`log`.*host_name*.

-o

Optional. Turn off temporary timing settings and reset them to the daemon starting state. The timing level is reset back to the value of the parameter for the corresponding daemon (LSB_TIME_MBD, LSB_TIME_SBD).

The log file is also reset back to the default log file.

host_name ...

Sets the timing level on the specified host or hosts.

Lists of hosts must be separated by spaces and enclosed in quotation marks.

Default: local host (host from which command was submitted)

mbdtime [**-l** *timing_level*] [**-f** *logfile_name*] [**-o**]

Sets timing level for MBD to include additional timing information in log files. You must be `root` or the LSF administrator to use this command.

See `sbdtime` for an explanation of options.

SEE ALSO

`bqueues(1), bhosts(1), lsb.queues(5), lsb.hosts(5), lsf.conf(5), lsf.cluster(5), sbatchd(8), mbatchd(8)`

bbot

moves a pending job relative to the last job in the queue

SYNOPSIS

bbot *job_ID* | "*job_ID*[*index_list*]" [*position*]

bbot [**-h** | **-V**]

DESCRIPTION

Changes the queue position of a pending job, or a pending job array element, to affect the order in which jobs are considered for dispatch.

By default, LSF dispatches jobs in a queue in the order of arrival (that is, first-come-first-served), subject to availability of suitable server hosts.

The `bbot` command allows users and the LSF administrator to manually change the order in which jobs are considered for dispatch. Users can only operate on their own jobs, whereas the LSF administrator can operate on any user's jobs. Users can only change the relative position of their own jobs.

If invoked by the LSF administrator, `bbot` moves the selected job after the last job with the same priority submitted to the queue. The positions of all users' jobs in the queue can be changed by the LSF administrator.

If invoked by a regular user, `bbot` moves the selected job after the last job with the same priority submitted by the user to the queue.

Pending jobs are displayed by `bjobs` in the order in which they will be considered for dispatch.

A user may use `bbot` to change the dispatch order of their jobs scheduled using a fairshare policy. However, if a job scheduled using a fairshare policy is moved by the LSF administrator using `btop`, the job will not be subject to further fairshare scheduling unless the same job is subsequently moved by the LSF administrator using `bbot`; in this case the job will be scheduled again using the same fairshare policy (see the `FAIRSHARE` keyword in `lsb.queues`(5) and `HostPartition` keyword in `lsb.hosts`(5)).

OPTIONS

job_ID | "job_ID[index_list]"

Required. Job ID of the job or job array on which to operate.

For a job array, the index list, the square brackets, and the quotation marks are required. An index list is used to operate on a job array. The index list is a comma separated list whose elements have the syntax *start_index*[*-end_index*[**:***step*]] where *start_index*, *end_index* and *step* are positive integers. If the step is omitted, a step of one is assumed. The job array index starts at one. The maximum job array index is 1000. All jobs in the array share the same job_ID and parameters. Each element of the array is distinguished by its array index.

position

Optional. The *position* argument can be specified to indicate where in the queue the job is to be placed. *position* is a positive number that indicates the target position of the job from the end of the queue. The positions are relative to only the applicable jobs in the queue, depending on whether the invoker is a regular user or the LSF administrator. The default value of 1 means the position is after all other jobs with the same priority.

-h

Prints command usage to stderr and exits.

-v

Prints LSF release version to stderr and exits.

SEE ALSO

bjobs(1), bswitch(1), btop(1)

bchkpnt

checkpoints one or more checkpointable jobs

SYNOPSIS

bchkpnt [**-f**] [**-k**] [**-p** *minutes* | **-p 0**] [*job_ID* | "*job_ID*[*index_list*]"] ...

bchkpnt [**-f**] [**-k**] [**-p** *minutes* | **-p 0**] [**-J** *job_name*]
[**-m** *host_name* | **-m** *host_group*] [**-q** *queue_name*]
[**-u** "*user_name*" | **-u all**] [**0**]

bchkpnt [**-h** | **-V**]

DESCRIPTION

Checkpoints your running (RUN) or suspended (SSUSP, USUSP, and PSUSP) checkpointable jobs. LSF administrators and root can checkpoint jobs submitted by other users.

By default, checkpoints one job, the most recently submitted job, or the most recently submitted job that also satisfies other specified options (`-m`, `-q`, `-u` and `-J`). Specify `-0` to checkpoint multiple jobs. Specify a job ID to checkpoint one specific job.

By default, jobs continue to execute after they have been checkpointed.

To submit a checkpointable job, use `bsub -k` or submit the job to a checkpoint queue (CHKPNT in `lsb.queues`(5)). Use `brestart(1)` to start checkpointed jobs.

LSF invokes the `echkpnt`(8) executable found in LSF_SERVERDIR to perform the checkpoint.

OPTIONS

`0`

(Zero). Checkpoints multiple jobs. Checkpoints all the jobs that satisfy other specified options (`-m`, `-q`, `-u` and `-J`).

`-f`

Forces a job to be checkpointed even if non-checkpointable conditions exist (these conditions are OS-specific).

-k

Kills a job after it has been successfully checkpointed.

-p *minutes* | **-p 0**

Enables periodic checkpointing and specifies the checkpoint period, or modifies the checkpoint period of a checkpointed job. Specify **-p 0** to disable periodic checkpointing.

Checkpointing is a resource-intensive operation. To allow your job to make progress while still providing fault tolerance, specify a checkpoint period of 30 minutes or longer.

-J *job_name*

Only checkpoints jobs that have the specified job name.

-m *host_name* | **-m** *host_group*

Only checkpoints jobs dispatched to the specified hosts.

-q *queue_name*

Only checkpoints jobs dispatched from the specified queue.

-u "*user_name*" | **-u all**

Only checkpoints jobs submitted by the specified users. The keyword **all** specifies all users. Ignored if a job ID other than 0 is specified.

job_ID | "*job_ID*[*index_list*]"

Checkpoints only the specified jobs.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

EXAMPLES

% bchkpnt 1234

Checkpoints the job with job ID 1234.

% bchkpnt -p 120 1234

Enables periodic checkpointing or changes the checkpoint period to 120 minutes (2 hours) for a job with job ID 1234.

```
% bchkpnt -m hostA -k -u all 0
```

When issued by root or an LSF administrator, will checkpoint and kill all checkpointable jobs on hostA. This is useful when a host needs to be shut down or rebooted.

SEE ALSO

`bsub(1), bmod(1), brestart(1), bjobs(1), bqueues(1), bhosts(1), libckpt.a(3), lsb.queues(5), echkpnt(8), erestart(8), mbatchd(8)`

bclusters

displays information about local MultiCluster queues

SYNOPSIS

bclusters [**-h** | **-V**]

DESCRIPTION

Displays a list of MultiCluster queues together with their relationship with queues in remote clusters.

OPTIONS

-h

Prints command usage to stderr and exits.

-v

Prints LSF release version to stderr and exits.

OUTPUT

LOCAL_QUEUE

Name of a local MultiCluster submission or execution queue.

JOB_FLOW

Indicates direction of job flow.

send

Means the local queue is a MultiCluster submission queue (SNDJOBS_TO is defined in the local queue).

recv

Means the local queue is a MultiCluster execution queue (RCVJOBS_FROM is defined in the local queue).

REMOTE

For submission queues, name of the execution queue in a remote cluster.

For execution queues, always "-".

CLUSTER

For submission queues, name of the remote cluster containing the execution queue.

For execution queues, name of the remote cluster that can send jobs to the local queue.

STATUS

Indicates the connection status between the local queue and remote queue.

ok

Means the two clusters can exchange information and the system is properly configured.

disc

Means communication between the two clusters has not been established. This could occur because there are no jobs waiting to be dispatched, or because the remote master cannot be located.

reject

Means the two clusters communicate but the remote queue does not accept jobs from the local queue.

FILES

Reads `lsb.queues`.

SEE ALSO

`lsfintro(1), lsclusters(1), ls_info(3), ls_policy(3), ls_clusterinfo(3), lsb.queues(5), lsb.queues(5)`

bhist

displays historical information about jobs

SYNOPSIS

bhist [**-a** | **-d** | **-p** | **-r** | **-s**] [**-b** | **-w**] [**-l**] [**-t**] [**-C** *time0*,*time1*] [**-D** *time0*,*time1*] [**-S** *time0*,*time1*] [**-T** *time0*,*time1*] [**-f** *logfile_name* | **-n** *number_logfiles* | **-n 0**] [**-J** *job_name*] [**-m** *host_name*] [**-N** *host_name* | **-N** *host_model* | **-N** *CPU_factor*] [**-P** *project_name*] [**-q** *queue_name*] [**-u** *user_name* | **-u all**]

bhist [**-J** *job_name*] [**-N** *host_name* | **-N** *host_model* | **-N** *CPU_factor*] [*job_ID ...* | "*job_ID***[***index***]**" ...]

bhist [**-h** | **-V**]

DESCRIPTION

Displays historical information about LSF jobs.

By default, displays information about your own pending, running and suspended jobs. By default, groups information by job. By default, CPU time is not normalized. By default, searches the event log file currently used by the LSF system: `$LSB_SHAREDIR/cluster_name/logdir/lsb.events` (see `lsb.events`(5)). By default, displays events occurring in the past week, but this can be changed by setting the environment variable LSB_BHIST_HOURS to an alternate number of hours.

If neither `-l` nor `-b` is present, the default is to display the fields in OUTPUT only (see below).

OPTIONS

`-a`

Displays information about both finished and unfinished jobs.

This option overrides `-d`, `-p`, `-s`, and `-r`.

`-b`

Brief format. Displays the information in a brief format. If used with the `-s` option, shows the reason why each job was suspended.

-d

Only displays information about finished jobs.

-l

Long format. Displays additional information. If used with `-s`, shows the reason why each job was suspended.

-p

Only displays information about pending jobs.

-r

Only displays information about running jobs.

-s

Only displays information about suspended jobs.

-t

Displays job events chronologically.

-w

Wide format. Displays the information in a wide format.

-C *time0,time1*

Only displays jobs that completed or exited during the specified time interval.

-D *time0,time1*

Only displays jobs dispatched during the specified time interval.

-S *time0,time1*

Only displays information about jobs submitted during the specified time interval.

-T *time0,time1*

Used together with `-t`.

Only displays information about job events within the specified time interval.

-f *logfile_name*

Searches the specified event log. Specify either an absolute or a relative path.

Useful for analysis directly on the file.

-J *job_name*

Only displays the jobs that have the specified *job_name*.

-m *host_name*

Only displays jobs dispatched to the specified host.

-n *number_logfiles* | **-n 0**

Searches the specified number of event logs, starting with the current event log and working through the most recent logs. Specify 0 to specify all the event log files in `$(LSB_SHAREDIR)/`*`cluster_name`*`/logdir`.

For example, if you specify 2, LSF searches `lsb.events` and `lsb.events.1`. If you specify 3, LSF searches `lsb.events`, `lsb.events.1`, and `lsb.events.2`.

-N *host_name* | **-N** *host_model* | **-N** *CPU_factor*

Normalizes CPU time by the specified CPU factor, or by the CPU factor of the specified host or host model.

If you use `bhist` directly on an event log, you must specify a CPU factor.

Use `lsinfo` to get host model and CPU factor information.

-P *project_name*

Only displays information about jobs belonging to the specified project.

-q *queue_name*

Only displays information about jobs submitted to the specified queue.

-u *user_name* | **-u all**

Displays information about jobs submitted by the specified user, or by all users if the keyword `all` is specified.

job_ID | "job_ID[*index*]"

Searches all event log files and only displays information about the specified jobs. If you specify a job array, displays all elements chronologically.

This option overrides all other options except `-J`, `-N`, `-h`, and `-V`. When it is used with `-J`, only those jobs listed here that have the specified job name are displayed.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

OUTPUT

Default Format

Statistics of the amount of time that a job has spent in various states:

PEND

The total waiting time excluding user suspended time before the job is dispatched.

PSUSP

The total user suspended time of a pending job.

RUN

The total run time of the job.

USUSP

The total user suspended time after the job is dispatched.

SSUSP

The total system suspended time after the job is dispatched.

UNKWN

The total unknown time of the job (job status becomes unknown if SBD on the execution host is temporarily unreachable).

TOTAL

The total time that the job has spent in all states; for a finished job, it is the turnaround time (that is, the time interval from job submission to job completion).

Long Format (-l)

Detailed history includes the date and time the job was forwarded and the name of the cluster to which the job was forwarded.

FILES

Reads `lsb.events`.

SEE ALSO

`lsb.events(5), bsub(1), bjobs(1), lsinfo(1)`

TIME FORMAT

The *time0*,*time1* in options of `-C`, `-S`, `-D`, and `-T` must conform to the following:

```
time_form = -ptime,ptime | ptime, | ,ptime | itime
ptime     = day | /day | month/ | year/month/day |
            year/month/day/ | hour: | month/day |
            year/month/day/hour: |
            year/month/day/hour:minute | day/hour: |
            month/day/hour: | day/hour:minute |
            hour:minute | month/day/hour:minute | . |
            .-itime
itime     = ptime
day, month, hour, minute = two digits
```

where:

- `ptime` stands for a specific point of time
- `itime` stands for a specific interval of time
- `.` (period) stands for the current month/day/hour:minute

Keeping the following rules in mind will help you to specify the time freely:

- year must be 4 digits and followed by a /
- month must be followed by a /
- day must be preceded by a /
- hour must be followed by a :

- minute must be preceded by a :
- The / before day can be omitted when day stands alone or when day is followed by /hour:
- No spaces are allowed in time format, that is, the time must be a single string.

The above time format is designed for easy and flexible time specification.

For example, suppose the current time is Mar 9 17:06:30 2000:

```
1,8       Mar 1 00:00:00 2000 to Mar 8 23:59:00 2000;
,4 or ,/4 the time when first job was logged to Mar 4
           23:59:00 2000;
6 or /6   Mar 6 00:00:00 2000 to Mar 6 23:59:00 2000;
2/        Feb 1 00:00:00 2000 to Feb 28 23:59:00 2000;
12:       Mar 9 12:00:00 2000 to Mar 9 12:59:00 2000;
2/1       Feb 1 00:00:00 2000 to Feb 1 23:59:00 2000;
2/1,      Feb 1 00:00:00 to the current time;
,. or ,   the time when first job was logged to the
          current time;
.-9,      Feb 28 17:06:30 2000 to Mar 9 17:06:30;
,.-2      the time when first job was logged to Mar 7
          17:06:30 2000;
,.-2/     the time when first job was logged to Jan 9
          17:06:30 2000;
,2/10:    the time when first job was logged to Mar 2
          10:59:00 2000;
1993/11/25,2000/1/25  from Nov 25 00:00:00 1993 to Jan
          25 23:59:00 2000
```

bhosts

displays hosts and their static and dynamic resources

SYNOPSIS

bhosts [**-w** | **-l**] [**-R** "*res_req*"] [*host_name* | *host_group*] ...

bhosts -s [*shared_resource_name* ...]

bhosts [**-h** | **-V**]

DESCRIPTION

Displays information about hosts.

By default, returns the following information about all hosts: host name, host status, job slot limits, and job state statistics.

The `-s` option displays information about the numeric shared resources and their associated hosts.

OPTIONS

`-w`

Displays host information in wide format. Fields are displayed without truncation.

`-l`

Displays host information in a (long) multi-line format. In addition to the default fields, displays information about the CPU factor, the dispatch windows, the current load, and the load thresholds.

`-R "`*`res_req`*`"`

Only displays information about hosts that satisfy the resource requirement expression. For more information about resource requirements, see `lsfintro(1)`. The size of the resource requirement string is limited to 512 bytes.

LSF supports ordering of resource requirements on all load indices, including external load indices, either static or dynamic.

host_name ... | host_group ...

Only displays information about the specified hosts or host groups. For host groups, the names of the hosts belonging to the group are displayed instead of the name of the host group. Do not use quotes when specifying multiple hosts or host groups.

-s [*shared_resource_name ...*]

Displays information about the specified shared resources. The resources must have numeric values. Returns the following information: the resource names, the total and reserved amounts, and the resource locations. If no shared resources are specified, displays information about all numeric shared resources.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

OUTPUT

Host-Based Default

Displays the following fields:

HOST_NAME

The name of the host. If a host has batch jobs running and the host is removed from the configuration, the host name will be displayed as `lost_and_found`.

STATUS

The current status of the host. Batch jobs can only be dispatched to hosts with an `ok` status. The possible values for host status are as follows:

ok

The host is available to accept batch jobs.

unavail

The host is down, or LIM and SBD on the host are unreachable.

unreach

LIM on the host is running but SBD is unreachable.

closed

The host is not allowed to accept any remote batch jobs. There are several reasons for the host to be closed (see Host-Based `-l` Options).

JL/U

The maximum number of job slots that the host can process on a per user basis.

For non-preemptive scheduling, these job slots are used by running jobs, as well as by suspended or pending jobs that have slots reserved for them.

For preemptive scheduling, these job slots are used by running jobs and by pending jobs that have slots reserved for them (see the description of PREEMPTIVE in `lsb.queues`(5) and JL/U in `lsb.hosts(5)`). If a dash (-) is displayed, there is no limit.

MAX

The maximum number of job slots that the host can process. These job slots are used by running and suspended jobs on the host, and by pending jobs that have jobs slots reserved for them on the host.

If preemptive scheduling is used, suspended jobs are not counted (see the description of PREEMPTIVE in `lsb.queues(5)` and MXJ in `lsb.hosts(5)`). If a dash (-) is displayed, there is no limit.

NJOBS

The number of job slots used by started jobs on the host (including running, suspended, or chunk jobs).

RUN

The number of job slots used by running jobs on the host.

SSUSP

The number of job slots used by system suspended jobs on the host.

USUSP

The number of job slots used by user suspended jobs on the host. Jobs can be suspended by the user or by the LSF administrator.

RSV

The number of job slots used by pending jobs that have jobs slots reserved on the host.

Host-Based -l Option

In addition to the above fields, the `-l` option also displays the following:

STATUS

closed

The long format shown by the `-l` option gives the possible reasons for a host to be closed:

closed_Adm

The host is closed by the LSF administrator or `root` (see `badmin(8)`). No job can be dispatched to the host, but jobs that are executing on the host will not be affected.

closed_Lock

The host is locked by the LSF administrator or `root` (see `lsadmin(8)`). All batch jobs on the host are suspended by LSF.

closed_Wind

The host is closed by its dispatch windows, which are defined in the configuration file `lsb.hosts(5)`. All batch jobs on the host are suspended by the LSF system.

closed_Full

The configured maximum number of batch job slots on the host has been reached (see MAX field below).

closed_Excl

The host is currently running an exclusive job.

closed_Busy

The host is overloaded, because some load indices go beyond the configured thresholds (see `lsb.hosts(5)`). The displayed thresholds that cause the host to be busy are preceded by an asterisk (*).

closed_LIM

LIM on the host is unreachable, but SBD is ok.

CPUF

Displays the CPU normalization factor of the host (see `lshosts(1)`).

DISPATCH_WINDOWS

Displays the dispatch windows for each host. The dispatch windows are the time windows during the week when batch jobs can be run on each host. Jobs already started are not affected by the dispatch windows. The default for the dispatch window is no restriction or always open (that is, twenty-four hours a day and seven days a week). For the dispatch window specification, see the description for the DISPATCH_WINDOWS keyword under the `-l` option in `bqueues(1)`.

CURRENT LOAD

Displays the total and reserved host load.

Reserved

You specify reserved resources by using `bsub -R` (see `lsfintro(1)`). These resources are reserved by jobs running on the host.

Total

The total load has different meanings depending on whether the load index is increasing or decreasing.

For increasing load indices, such as run queue lengths, CPU utilization, paging activity, logins, and disk I/O, the total load is the consumed plus the reserved amount. The total load is calculated as the sum of the current load and the reserved load. The current load is the load seen by `lsload(1)`.

For decreasing load indices, such as available memory, idle time, available swap space, and available space in tmp, the total load is the available amount. The total load is the difference between the current load and the reserved load. This difference is the available resource as seen by `lsload(1)`.

LOAD THRESHOLD

Displays the scheduling threshold `loadSched` and the suspending threshold `loadStop`. Also displays the migration threshold if defined and the checkpoint support if the host supports checkpointing.

The format for the thresholds is the same as for batch job queues (see `bqueues(1)`) and `lsb.queues(5)`). For an explanation of the thresholds and load indices, see the description for the "QUEUE SCHEDULING PARAMETERS" keyword under the `-l` option in `bqueues(1)`.

Resource-Based -s Option

The -s option displays the following: the amounts used for scheduling, the amounts reserved, and the associated hosts for the shared resources. Only shared resources with numeric values are displayed. See `lim(8)`, and `lsf.cluster(5)` on how to configure shared resources.

The following fields are displayed:

RESOURCE

The name of the resource.

TOTAL

The value of the shared resource used for scheduling. This is the sum of the current and the reserved load for the shared resource.

RESERVED

The amount reserved by jobs. You specify the reserved resource using `bsub -R` (see `lsfintro(1)`).

LOCATION

The hosts that are associated with the shared resource.

SEE ALSO

lsb.hosts(5), bqueues(1), lsfintro(1), lshosts(1), badmin(8), lsadmin(8)

bhpart

displays information about host partitions

SYNOPSIS

bhpart [**-r**] [*host_partition_name ...*]

bhpart [**-h** | **-V**]

DESCRIPTION

Displays information about host partitions. By default, displays information about all host partitions.

Host partitions are used to configure host-partition fairshare scheduling.

OPTIONS

-r

Displays the entire information tree associated with the host partition recursively.

host_partition_name ...

Displays information about the specified host partitions only.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

OUTPUT

The following fields are displayed for each host partition:

HOST_PARTITION_NAME

Name of the host partition.

HOSTS

Hosts or host groups that are members of the host partition. The name of a host group is appended by a slash (/) (see `bmgroup(1)`).

USER/GROUP

Name of users or user groups who have access to the host partition (see `bugroup(1)`).

SHARES

Number of shares of resources assigned to each user or user group in this host partition, as configured in the file `lsb.hosts`. The shares affect dynamic user priority for when fairshare scheduling is configured at the host level.

PRIORITY

Dynamic user priority for the user or user group. Larger values represent higher priorities. Jobs belonging to the user or user group with the highest priority are considered first for dispatch.

In general, users or user groups with larger SHARES, fewer STARTED and RESERVED, and a lower CPU_TIME and RUN_TIME will have higher PRIORITY.

STARTED

Number of job slots used by running or suspended jobs owned by users or user groups in the host partition.

RESERVED

Number of job slots reserved by the jobs owned by users or user groups in the host partition.

CPU_TIME

Cumulative CPU time used by jobs of users or user groups executed in the host partition. Measured in seconds, to one decimal place.

LSF calculates the cumulative CPU time using the actual (not normalized) CPU time and a decay factor such that 1 hour of recently-used CPU time decays to 0.1 hours after an interval of time specified by HIST_HOURS in `lsb.params` (5 hours by default).

RUN_TIME

Wall-clock run time plus historical run time of jobs of users or user groups that are executed in the host partition. Measured in seconds.

LSF calculates the historical run time using the actual run time of finished jobs and a decay factor such that 1 hour of recently-used run time decays to 0.1 hours after an interval of time specified by HIST_HOURS in `lsb.params` (5 hours by default). Wall-clock run time is the run time of running jobs.

FILES

Reads `lsb.hosts`.

SEE ALSO

`bugroup(1), bmgroup(1), lsb.hosts(5)`

bjobs

displays information about LSF jobs

SYNOPSIS

bjobs [**-a**] [**-A**] [**-g**] [**-R**] [**-w** | **-l**] [**-J** *job_name*]
[**-m** *host_name* | **-m** *host_group*]
[**-N** *host_name* | **-N** *host_model* | **-N** *CPU_factor*] [**-P** *project_name*]
[**-q** *queue_name*] [**-u** *user_name* | **-u** *user_group* | **-u all**] *job_ID ...*

bjobs [**-d**] [**-p**] [**-r**] [**-s**] [**-A**] [**-g**] [**-R**] [**-w** | **-l**] [**-J** *job_name*]
[**-m** *host_name* | **-m** *host_group*]
[**-N** *host_name* | **-N** *host_model* | **-N** *CPU_factor*] [**-P** *project_name*]
[**-q** *queue_name*] [**-u** *user_name* | **-u** *user_group* | **-u all**] *job_ID ...*

bjobs [**-h** | **-V**]

DESCRIPTION

Displays information about jobs.

By default, displays information about your own pending, running and suspended jobs.

To display older historical information, use `bhist`.

OPTIONS

`-a`

Displays information about jobs in all states, including finished jobs that finished recently, within an interval specified by CLEAN_PERIOD in `lsb.params` (the default period is 1 hour).

`-A`

Displays summarized information about job arrays. If you specify job arrays with the job array ID, and also specify `-A`, do not include the index list with the job array ID.

You can use `-w` to show the full array specification, if necessary.

-d

Displays information about jobs that finished recently, within an interval specified by CLEAN_PERIOD in `lsb.params` (the default period is 1 hour).

-g

Displays information about job groups.

-p

Displays pending jobs, together with the pending reasons that caused each job not to be dispatched during the last dispatch turn. The pending reason shows the number of hosts for that reason, or names the hosts if `-l` is also specified.

Each pending reason is associated with one or more hosts and it states the cause why these hosts are not allocated to run the job. In situations where the job requests specific hosts (using `bsub -m`), users may see reasons for unrelated hosts also being displayed, together with the reasons associated with the requested hosts. The life cycle of a pending reason ends after a new dispatch turn starts. The reason may not reflect the current load situation because it could last as long as the interval specified by MBD_SLEEP_TIME in `lsb.params`.

When the job slot limit is reached for a job array (`bsub -J "jobArray[indexList]%job_slot_limit"`) the following message is displayed:

```
The job array has reached its job slot limit.
```

-r

Displays running jobs.

-R

Displays jobs in a job group tree recursively.

-s

Displays suspended jobs, together with the suspending reason that caused each job to become suspended.

The suspending reason may not remain the same while the job stays suspended. For example, a job may have been suspended due to the paging rate, but after the paging rate dropped another load index could prevent the job from being resumed. The suspending reason will be

updated according to the load index. The reasons could be as old as the time interval specified by SBD_SLEEP_TIME in `lsb.params`. So the reasons shown may not reflect the current load situation.

-w

Wide format. Displays job information without truncating fields.

-l

Long format. Displays detailed information for each job in a multi-line format.

The `-l` option displays the following additional information: project name, job command, current working directory on the submission host, pending and suspending reasons, job status, resource usage, resource limits, job group information.

Use `bjobs -A -l` to display detailed information for job arrays including job array job limit (*`%job_limit`*) if set.

job_ID

Displays information about the specified jobs or job arrays.

If you use `-A`, specify job array IDs without the index list.

-J *job_name*

Displays information about the specified jobs or job arrays.

-m *host_name* | **-m** *host_group*

Only displays jobs dispatched to the specified hosts.

To determine the available hosts and host groups, use `bhosts` and `bmgroup`.

-N *host_name* | **-N** *host_model* | **-N** *CPU_factor*

Displays the normalized CPU time consumed by the job. Normalizes using the CPU factor specified, or the CPU factor of the host or host model specified.

-P *project_name*

Only displays jobs that belong to the specified project.

-q *queue_name*

Only displays jobs in the specified queue.

The command **bqueues** returns a list of queues configured in the system, and information about the configurations of these queues.

-u *user_name* | **-u** *user_group* | **-u all**

Only displays jobs that have been submitted by the specified users. The keyword **all** specifies all users.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

OUTPUT

Pending jobs are displayed in the order in which they will be considered for dispatch. Jobs in higher priority queues are displayed before those in lower priority queues. Pending jobs in the same priority queues are displayed in the order in which they were submitted but this order can be changed by using the commands **btop** or **bbot**. If more than one job is dispatched to a host, the jobs on that host are listed in the order in which they will be considered for scheduling on this host by their queue priorities and dispatch times. Finished jobs are displayed in the order in which they were completed.

Default Display

A listing of jobs is displayed with the following fields:

JOBID

The job ID that LSF assigned to the job.

USER

The user who submitted the job.

STAT

The current status of the job (see JOB STATUS below).

QUEUE

The name of the job queue to which the job belongs. If the queue to which the job belongs has been removed from the configuration, the queue name will be displayed as **lost_and_found**. Use **bhist**

to get the original queue name. The job in the `lost_and_found` queue will remain pending until it is switched with the `bswitch` command into another queue.

FROM_HOST

The name of the host from which the job was submitted.

EXEC_HOST

The name of one or more hosts on which the job is executing (this field is empty if the job has not been dispatched). If the host on which the job is running has been removed from the configuration, the host name will be displayed as `lost_and_found`. Use `bhist` to get the original host name.

JOB_NAME

The job name assigned by the user, or the *command* string assigned by default (see `bsub` (1)). If the job name is too long to fit in this field, then only the latter part of the job name is displayed.

SUBMIT_TIME

The submission time of the job.

-l output

If the `-l` option is specified, the resulting long format listing includes the following additional fields:

Project

The project the job was submitted from.

Command

The job command.

CWD

The current working directory on the submission host.

PENDING REASONS

The reason the job is in the PEND or PSUSP state. The names of the hosts associated with each reason will be displayed when both `-p` and `-l` options are specified.

SUSPENDING REASONS

The reason the job is in the USUSP or SSUSP state.

loadSched

The load scheduling thresholds for the job.

loadStop

The load suspending thresholds for the job.

JOB STATUS

Possible values for the status of a job include:

PEND

The job is pending, that is, it has not yet been started.

PSUSP

The job has been suspended, either by its owner or the LSF administrator, while pending.

RUN

the job is currently running.

USUSP

The job has been suspended, either by its owner or the LSF administrator, while running.

SSUSP

The job has been suspended by LSF. The job has been suspended by LSF due to either of the following two causes:

1) The load conditions on the execution host or hosts have exceeded a threshold according to the `loadStop` vector defined for the host or queue.

2) the run window of the job's queue is closed. See `bqueues(1)`, `bhosts(1)`, and `lsb.queues(5)`.

DONE

The job has terminated with status of 0.

EXIT

The job has terminated with a non-zero status – it may have been aborted due to an error in its execution, or killed by its owner or the LSF administrator.

UNKWN

MBD has lost contact with the SBD on the host on which the job runs.

WAIT

For jobs submitted to a chunk job queue, members of a chunk job that are waiting to run.

ZOMBI

A job will become ZOMBI if:

- A non-rerunnable job is killed by `bkill` while the SBD on the execution host is unreachable and the job is shown as UNKWN.

- The host on which a rerunnable job is running is unavailable and the job has been requeued by LSF with a new job ID, as if the job were submitted as a new job.

After the execution host becomes available, LSF will try to kill the ZOMBI job. Upon successful termination of the ZOMBI job, the job's status will be changed to EXIT.

RESOURCE USAGE

The values for the current usage of a job include:

CPU time

Cumulative total CPU time in seconds of all processes in a job.

MEM

Total resident memory usage of all processes in a job, in MB.

SWAP

Total virtual memory usage of all processes in a job, in MB.

PGID

Currently active process group ID in a job.

PIDs

Currently active processes in a job.

RESOURCE LIMITS

The hard resource limits that are imposed on the jobs in the queue (see `getrlimit`(2) and `lsb.queues`(5)). These limits are imposed on a per-job and a per-process basis.

The possible per-job limits are:

CPULIMIT

PROCLIMIT

MEMLIMIT

SWAPLIMIT

PROCESSLIMIT

The possible UNIX per-process resource limits are:

RUNLIMIT

FILELIMIT

DATALIMIT

STACKLIMIT

CORELIMIT

If a job submitted to the queue has any of these limits specified (see `bsub`(1)), then the lower of the corresponding job limits and queue limits are used for the job.

If no resource limit is specified, the resource is assumed to be unlimited.

Job Group Information

For job groups, the parent job group, together with the following fields are displayed:

GROUP

Job group name.

STAT

Job group status. Possible values for the job group status are listed as follows:

HOLD

The job group is on hold, either by the parent or ancestor job groups or by the group owner or the LSF administrator.

ACTIVE

The job group is active.

INACTIVE

The job group is inactive.

OWNER

Job group owner.

NJOBS

Total number of jobs in the job group.

PEND, DONE, RUN, EXIT, SSUSP, USUSP, PSUSP

The number of jobs of the group in PEND, DONE, RUN, EXIT, SUSUP, USUSP and PSUSP status respectively.

Job Array Summary Information

If you use `-A`, displays summary information about job arrays. The following fields are displayed:

JOBID

Job ID of the job array.

ARRAY_SPEC

Array specification in the format of *`name`*`[`*`index`*`]`. The array specification may be truncated, use `-w` option together with `-A` to show the full array specification.

OWNER

Owner of the job array.

NJOBS

Number of jobs in the job array.

PEND

Number of pending jobs of the job array.

RUN

Number of running jobs of the job array.

DONE

Number of successfully completed jobs of the job array.

EXIT

Number of unsuccessfully completed jobs of the job array.

SSUSP

Number of LSF system suspended jobs of the job array.

USUSP

Number of user suspended jobs of the job array.

PSUSP

Number of held jobs of the job array.

EXAMPLES

% bjobs -pl

Displays detailed information about all pending jobs of the invoker.

% bjobs -ps

Display only pending and suspended jobs.

% bjobs -u all -a

Displays all jobs of all users.

% bjobs -d -q short -m apple -u john

Displays all the recently finished jobs submitted by john to the queue `short`, and executed on the host `apple`.

% bjobs 101 102 203 509

Display jobs with job_ID 101, 102, 203, and 509.

SEE ALSO

`bsub(1), bkill(1), bhosts(1), bmgroup(1), bqueues(1) bhist(1), bresume(1), bdel(1), bstop(1), lsb.params(5), mbatchd(8)`

bkill

sends signals to kill, suspend, or resume unfinished jobs

SYNOPSIS

bkill [**-l**] [**-R**] [**-J** *job_name*] [**-m** *host_name* | **-m** *host_group*]
[**-q** *queue_name*] [**-r** | **-s (***signal_value* | *signal_name***)**]
[**-u** *user_name* | **-u** *user_group* | **-u all**]
[*job_ID* ... | **0** | "*job_ID***[***index***]**" ...]

bkill [**-h** | **-V**]

DESCRIPTION

By default, sends a set of signals to kill the specified jobs. On UNIX, SIGINT and SIGTERM are sent to give the job a chance to clean up before termination, then SIGKILL is sent to kill the job. The time interval between sending each signal is defined by the JOB_TERMINATE_INTERVAL parameter in `lsb.params(5)`.

On Windows NT, job control messages replace the SIGINT and SIGTERM signals (but only customized applications can process them) and the `TerminateProcess()` system call is sent to kill the job.

Users can only operate on their own jobs. Only `root` and LSF administrators can operate on jobs submitted by other users.

If a signal request fails to reach the job execution host, LSF tries the operation later when the host becomes reachable. LSF retries the most recent signal request.

If a job is running in a queue with CHUNK_JOB_SIZE set, `bkill` has the following results depending on job state:

PEND

Job is removed from chunk (NJOBS -1, PEND -1)

RUN

All jobs in the chunk are suspended (NRUN -1, NSUSP +1)

USUSP

Job finishes, next job in the chunk starts if one exists (NJOBS -1, PEND -1, SUSP -1, RUN +1)

WAIT

Job finishes (NJOBS-1, PEND -1)

To remove a repetitive job from the system, use **bdel**. Using **bkill** on a repetitive job kills the current run, if the job has been started, and requeues the job. See **bcadd**(1) and **bsub**(1) for information on setting up a job to run repetitively. See **bdel(1)** for more information.

If the job cannot be killed, use **bkill -r** to remove the job from the LSF system without waiting for the job to terminate, and free the resources of the job.

A job ID or **-m**, **-u**, **-q**, or **-J** must be specified with **bkill**.

OPTIONS

-l

Displays the signal names supported by **bkill**. This is a subset of signals supported by **/bin/kill** and is platform-dependent.

-R

Operates on jobs recursively on the job group tree.

-J *job_name*

Operates only on jobs with the specified *job_name*. The **-J** option is ignored if a job ID other than 0 is specified in the *job_ID* option.

-m *host_name* | **-m** *host_group*

Operates only on jobs dispatched to the specified host or host group.

If *job_ID* is not specified, only the most recently submitted qualifying job is operated on. The **-m** option is ignored if a job ID other than 0 is specified in the *job_ID* option. See **bhosts(1)** and **bmgroup(1)** for more information about hosts and host groups.

-q *queue_name*

Operates only on jobs in the specified queue.

If *job_ID* is not specified, only the most recently submitted qualifying job is operated on.

The **-q** option is ignored if a job ID other than 0 is specified in the *job_ID* option.

See **bqueues**(1) for more information about queues.

-r

Removes a job from the LSF system without waiting for the job to terminate in the operating system.

Sends the same series of signals as **bkill** without **-r**, except that the job is removed from the system immediately, the job is marked as EXIT, and the job resources that LSF monitors are released as soon as LSF receives the first signal.

Also operates on jobs for which a **bkill** command has been issued but which cannot be reached to be acted on by SBD (jobs in ZOMBI state). If SBD recovers before the jobs are completely removed, LSF ignores the zombi jobs killed with **bkill -r**.

Use **bkill -r** only on jobs that cannot be killed in the operating system, or on jobs that cannot be otherwise removed using the **bdel** or **bkill**.

The **-r** option cannot be used with the **-s** option.

-s (*signal_value* | *signal_name***)**

Sends the specified signal to specified jobs. You can specify either a name, stripped of the SIG prefix (such as KILL), or a number (such as 9).

Eligible signal names are listed by **bkill -l**.

The **-s** option cannot be used with the **-r** option.

Use **bkill -s** to suspend and resume jobs by using the appropriate signal instead of using **bstop** or **bresume**. Sending the SIGCONT signal is the same as using **bresume**. Sending the SIGSTOP signal to sequential jobs or the SIGTSTP to parallel jobs is the same as using **bstop**.

You cannot suspend a job that is already suspended, or resume a job that is not suspended. Using SIGSTOP or SIGTSTP on a job that is in the USUSP state has no effect and using SIGCONT on a job that is not in either the PSUSP or the USUSP state has no effect. See **bjobs(1)** for more information about job states.

-u *user_name* | **-u** *user_group* | **-u all**

Operates only on jobs submitted by the specified user or user group (see **bugroup(1)**), or by all users if the reserved user name **all** is specified.

If *job_ID* is not specified, only the most recently submitted qualifying job is operated on. The `-u` option is ignored if a job ID other than `0` is specified in the *job_ID* option.

job_ID ... | 0 | "job_ID[index]" ...

Operates only on jobs that are specified by *job_ID* or "*job_ID*[*index*]", where "*job_ID*[*index*]" specifies selected job array elements (see `bjobs(1)`). For job arrays, quotation marks must enclose the job ID and index, and index must be enclosed in square brackets.

Jobs submitted by any user can be specified here without using the `-u` option. If you use the reserved job ID 0, all the jobs that satisfy other options (that is, `-m`, `-q`, `-u` and `-J)` are operated on; all other job IDs are ignored.

The options `-u`, `-q`, `-m` and `-J` have no effect if a job ID other than 0 is specified. Job IDs are returned at job submission time (see `bsub(1)`) and may be obtained with the `bjobs` command (see `bjobs(1)`).

`-h`

Prints command usage to stderr and exits.

`-V`

Prints LSF release version to stderr and exits.

EXAMPLES

`% bkill -s 17 -q night`

Sends signal 17 to the last job that was submitted by the invoker to queue `night`.

`% bkill -q short -u all 0`

Kills all the jobs that are in the queue `short`.

`% bkill -r 1045`

Forces the removal of unkillable job 1045.

SEE ALSO

`bsub(1), bjobs(1), bqueues(1), bhosts(1), bresume(1), bstop(1), bdel(1), bparams(5), mbatchd(8), kill(1), signal(2)`

bmgroup

displays information about host groups

SYNOPSIS

bmgroup [**-r**] [**-w**] [*host_group* ...]

bmgroup [**-h** | **-V**]

DESCRIPTION

Displays host groups and host names for each group.

By default, displays information about all host groups. A host partition is also considered a host group.

OPTIONS

-r

Expands host groups recursively. The expanded list contains only host names; it does not contain the names of subgroups. Duplicate names are listed only once.

-w

Wide format. Displays host and host group names without truncating fields.

host_group ...

Only displays information about the specified host groups. Do not use quotes when specifying multiple host groups.

-h

Prints command usage to stderr and exits.

-v

Prints LSF release version to stderr and exits.

OUTPUT

In the list of hosts, a name followed by a slash (/) indicates a subgroup.

FILES

Host groups and host partitions are defined in the configuration file `lsb.hosts`(5).

SEE ALSO

`lsb.hosts(5), bugroup(1), bhosts(1)`

bmig

migrates checkpointable or rerunnable jobs

SYNOPSIS

bmig [**-f**] [*job_ID* | "*job_ID*[*index_list*]"] ...

bmig [**-f**] [**-J** *job_name*] [**-m** "*host_name* ..." | **-m** "*host_group* ..."] [**-u** *user_name* | **-u** *user_group* | **-u all**] [**0**]

bmig [**-h** | **-V**]

DESCRIPTION

Migrates one or more of your checkpointable and rerunnable jobs. LSF administrators and `root` can migrate jobs submitted by other users.

By default, migrates one job, the most recently submitted job, or the most recently submitted job that also satisfies other specified options (`-u` and `-J`). Specify `0` to migrate multiple jobs.

To migrate a job, both hosts must be binary compatible, run the same OS version, have access to the executable, have access to all open files (LSF must locate them with an absolute path name), and have access to the checkpoint directory.

Only started jobs can be migrated (i.e., running or suspended jobs); pending jobs cannot be migrated.

When a checkpointable job is migrated, LSF checkpoints and kills the job (similar to the `-k` option of `bchkpnt(1)`) then restarts it on the next available host. If checkpoint is not successful, the job is not killed and remains on the host. If a job is being checkpointed when `bmig` is issued, the migration is ignored. This situation may occur if periodic checkpointing is enabled.

When a rerunnable job is migrated, LSF kills the job (similar to `bkill(1)`) then restarts it from the beggining on the next available host.

The environment variable LSB_RESTART is set to Y when a migrating job is restarted or rerun.

A job is made rerunnable by specifying the `-r` option on the command line using `bsub(1)` and `bmod(1)`, or automatically by configuring RERUNNABLE in `lsb.queues(5)`.

A job is made checkpointable by specifying the location of a checkpoint directory on the command line using the `-k` option of `bsub(1)` and `bmod(1)`, or automatically by configuring CHKPNT in `lsb.queues(5)`.

OPTIONS

`-f`

Forces a checkpointable job to be checkpointed even if non-checkpointable conditions exist (these conditions are OS-specific).

`job_ID` **`|`** **`"`*** ***`job_ID`*** **`[`*** ***`index_list`*** **`]"`** **`|`** **`0`**

Specifies the job ID of the jobs to be migrated. The `-J` and `-u` options are ignored.

If you specify a job ID of **0**, all other job IDs are ignored, and all jobs that satisfy the `-J` and `-u` options are migrated.

If you do not specify a job ID, the most recently submitted job that satisfies the `-J` and `-u` options is migrated.

`-J` ***`job_name`***

Specifies the job name of the job to be migrated. Ignored if a job ID other than 0 is specified.

`-m` **`"`*** ***`host_name ...`*** **`"`** **`|`** **`-m`** **`"`*** ***`host_group ...`*** **`"`**

Only migrates jobs dispatched to the specified hosts.

`-u` **`"`*** ***`user_name`*** **`"`** **`|`** **`-u`** **`"`*** ***`user_group`*** **`"`** **`|`** **`-u all`**

Specifies that only jobs submitted by these users are to be migrated.

If the reserved user name `all` is specified, jobs submitted by all users are to be migrated. Ignored if a job ID other than 0 is specified.

`-h`

Prints command usage to stderr and exits.

`-V`

Prints LSF release version to stderr and exits.

SEE ALSO

bsub(1), brestart(1), bchkpnt(1), bjobs(1), bqueues(1), bhosts(1), bugroup(1), mbatchd(8), lsb.queues(5), kill(1)

bmod

modifies job submission options of a job

SYNOPSIS

bmod [*bsub options*] [*job_ID* **|** "*job_ID***[***index***]**"]

bmod [**-h** | **-V**]

OPTION LIST

[**-B** | **-Bn**]
[**-N** | **-Nn**]
[**-r** | **-rn**]
[**-x** | **-xn**]
[**-b** *begin_time* | **-bn**]
[**-C** *core_limit* | **-Cn**]
[-c [*hour***:**]*minute*[*/host_name* | */host_model*] | **-cn**]
[**-D** *data_limit* | **-Dn**]
[**-e** *err_file* | **-en**]
[**-E** "*pre_exec_command* [*argument* ...]" | **-En**]
[**-f** "*local_file op* [*remote_file*]" ... | **-fn**]
[**-F** *file_limit* | **-Fn**]
[**-G** *user_group* | **-Gn**]
[**-i** *input_file* | **-in** | **-is** *input_file* | **-isn**]
[**-J** *job_name* | **-J** "**%***job_limit*" | **-Jn**]
[**-k** *checkpoint_dir* | **-k** "*checkpoint_dir* [*checkpoint_period*]" | **-kn**]
[**-L** *login_shell* | **-Ln**]
[**-m** "*host_name*[**+**[*pref_level*]] | *host_group*[**+**[*pref_level*]] ..." | **-mn**]
[**-M** *mem_limit* | **-Mn**]
[**-n** *num_processors* | **-nn**]
[**-o** *out_file* | **-on**]
[**-P** *project_name* | **-Pn**]
[**-p** *process_limit* | **-Pn**]
[**-q** "*queue_name* ..." | **-qn**]
[**-R** "*res_req*" | **-Rn**]
[-**sp** *priority* | -**spn**]
[**-S** *stack_limit* | **-Sn**]
[**-t** *term_time* | **-tn**]
[**-u** *mail_user* | **-un**]
[**-w** '*dependency_expression*' | **-wn**]

[**-W** *run_limit* [*/host_name* | */host_model*] | **-Wn**]
[**-Z** "*new_command*" | **-Zs** "*new_command*" | **-Zsn**]
[*job_ID* | "*job_ID***[***index***]**"]
[**-h**]
[**-V**]

DESCRIPTION

Modifies the options of a previously submitted job. See `bsub(1)` for complete descriptions of job submission options you can modify with `bmod`.

Only the owner of the job, or an LSF administrator, can modify the options of a job.

All options specified at submission time may be changed. The value for each option may be overridden with a new value by specifying the option as in `bsub`. To reset an option to its default value, use the option string followed by '`n`'. Do not specify an option value when resetting an option.

The `-i`, `-in`, and `-Z` options have counterparts that support spooling of input and job command files (`-is`, `-isn`, `-Zs`, and `-Zsn`).

You can modify all options of a pending job, even if the corresponding `bsub` option was not specified. Modifying a job that is pending in a chunk job queue (CHUNK_JOB_SIZE) removes the job from the chunk to be scheduled later.

By default, you can modify resource reservation for running jobs (`-R "`*res_req*`"`). To modify additional job options for running jobs, define LSB_MOD_ALL_JOBS=Y in `lsf.conf`.

The following are the only `bmod` options that are valid for running jobs. You cannot make any other modifications after a job has been dispatched.

- Resource reservation (`-R "`*res_req*`"`)
- CPU limit (`-c` [*hour*`:`]*minute*[`/`*host_name* | `/`*host_model*])
- Memory limit (`-M` *mem_limit*)
- Run limit (`-W` *run_limit*[`/`*host_name* | `/`*host_model*])
- Standard output file name (`-o` *output_file*)
- Standard error file name (`-e` *error_file*)

- Rerunnable jobs (`-r | -rn`)

Modified resource limits cannot exceed the resource limits defined in the queue.

To modify the CPU limit or the memory limit of running jobs, the parameters LSB_JOB_CPULIMIT=Y and LSB_JOB_MEMLIMIT=Y must be defined in `lsf.conf`.

OPTIONS

***job_ID* | "*job_ID*[*index*]"**

Modifies jobs with the specified job ID.

Modifies job array elements specified by "*job_ID*[*index*]".

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

SEE ALSO

`bsub(1), bcadd(1), bcal(1), lsfjs(1), lsfbatch(1)`

LIMITATIONS

Modifying the `-q` option of a job array is not permitted.

Modifying remote running jobs in a MultiCluster environment is not supported.

If you do not specify `-e` before the job is dispatched, you cannot modify the name of job error file for a running job. Modifying the job output options of remote running jobs is not supported.

bparams

displays information about configurable system parameters in `lsb.params`

SYNOPSIS

bparams [**-l**]

bparams [**-h** | **-V**]

DESCRIPTION

Displays information about configurable system parameters in `lsb.params`.

By default, displays only the interesting parameters.

OPTIONS

`-l`

Long format. Displays detailed information about all the configurable parameters in `lsb.params`.

`-h`

Prints command usage to stderr and exits.

`-v`

Prints LSF release version to stderr and exits.

SEE ALSO

`lsb.params(5)`

bpeek

displays the stdout and stderr output of an unfinished job

SYNOPSIS

bpeek [**-f**] [**-q** *queue_name* | **-m** *host_name* | **-J** *job_name* | *job_ID* | "*job_ID*[*index_list*]"]

bpeek [**-h** | **-V**]

DESCRIPTION

Displays the standard output and standard error output that have been produced by one of your unfinished jobs, up to the time that this command is invoked.

By default, displays the output using the command `cat`.

This command is useful for monitoring the progress of a job and identifying errors. If errors are observed, valuable user time and system resources can be saved by terminating an erroneous job.

OPTIONS

`-f`

Displays the output of the job using the command `tail -f`.

`-q` *`queue_name`*

Operates on your most recently submitted job in the specified queue.

`-m` *`host_name`*

Operates on your most recently submitted job that has been dispatched to the specified host.

`-J` *`job_name`*

Operates on your most recently submitted job that has the specified job name.

`job_ID` | **`"`***`job_ID`***`[`***`index_list`***`]"`**

Operates on the specified job.

-h

Prints command usage to stderr and exits.

-v

Prints LSF release version to stderr and exits.

SEE ALSO

`tail(1), bsub(1), bjobs(1), bhist(1), bhosts(1), bqueues(1)`

bpost

sends messages and attaches data files to a job

SYNOPSIS

bpost [**-i** *message_index*] [**-d** "*description*"] [**-a** *data_file*] *job_ID* | "*job_ID*[*index_list*]" | **-J** *job_name*

bpost [**-h** | **-V**]

DESCRIPTION

Provides external status information or sends data to a job in the system. Done or exited jobs cannot accept messages.

By default, operates on the message index 0. By default, posts the message `"no description"`.

Users can only send messages and data to their own jobs. Root and LSF administrators can also send messages to jobs submitted by other users; they cannot attach data files to jobs owned by other users.

A job can accept messages until it is cleaned from the system. If your application requires transfer of data from one job to another, use the `-a` option of `bpost`(1) to attach a data file to the job, then use the `bread`(1) command to copy the attachment to another file.

You can associate several messages and attached data files with the same job. As the job is processed, use `bread`(1) or `bstatus`(1) to retrieve the messages posted to the job. Use `bread`(1) to copy message attachments to external files. You can also use `bstatus -d` to update the external job status.

The command:

```
$ bstatus -d "description"
```

is equivalent to:

```
$ bpost -i 0 -d "description"
```

OPTIONS

`-i` *`message_index`*

Operates on the specified message index.

-d *"description"*

Places your own status text as a message to the job. The message description has a maximum length of 512 characters.

For example, your application may require additional job status descriptions besides the ones that LSF provides internally (`PEND`, `RUN`, `SUSP`, etc.)

-a *data_file*

Attaches the specified data file to the job external storage.

Use the JOB_ATTA_DIR parameter in `lsb.params`(5) to specify the directory where attachment data files are saved. The directory must have at least 1 MB of free space. MBD checks for available space in the job attachment directory before transferring the file.

Use the MAX_JOB_ATTA_SIZE parameter in `lsb.params` to set a maximum size for job message attachments.

job_ID | *"job_ID*[*index_list*]*"* | **-J** *job_name*

Required. Specify the job to operate on.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

EXAMPLE

```
% bpost -i 1 -d "step 1" -a step1.out 2500
```

Puts the message text `step 1` into message index 1, and attaches the file `step1.out` to job 2500.

SEE ALSO

`bread(1)`, `bstatus(1)`, MAX_JOB_ATTA_SIZE, MAX_JOB_MSG_NUM

bqueues

displays information about queues

SYNOPSIS

bqueues [**-w** | **-l** | **-r**] [**-m** *host_name* | **-m** *host_group* | **-m all**] [**-u** *user_name* | **-u** *user_group* | **-u all**] [*queue_name* ...]

bqueues [**-h** | **-V**]

DESCRIPTION

Displays information about queues.

By default, returns the following information about all queues: queue name, queue priority, queue status, job slot statistics, and job state statistics.

Batch queue names and characteristics are set up by the LSF administrator (see `lsb.queues`(5) and `mbatchd(8)`).

OPTIONS

`-w`

Displays queue information in a wide format. Fields are displayed without truncation.

`-l`

Displays queue information in a long multi-line format. The `-l` option displays the following additional information: queue description, queue characteristics and statistics, scheduling parameters, resource limits, scheduling policies, users, hosts, user shares, windows, associated commands, and job controls.

`-r`

Displays the same information as the `-l` option. In addition, if fairshare is defined for the queue, displays recursively the share account tree of the fairshare queue.

-m *host_name* | **-m** *host_group* | **-m all**

Displays the queues that can run jobs on the specified host or host group. If the keyword `all` is specified, displays the queues that can run jobs on all hosts . For a list of host groups see `bmgroup(1)`.

-u *user_name* | **-u** *user_group* | **-u all**

Displays the queues that can accept jobs from the specified user or user group (For a list of user groups see `bugroup(1)`.) If the keyword 'all' is specified, displays the queues that can accept jobs from all users.

queue_name ...

Displays information about the specified queues.

-h

Prints command usage to stderr and exits.

-v

Prints LSF release version to stdout and exits.

OUTPUT

Default Output

Displays the following fields:

QUEUE_NAME

The name of the queue. Queues are named to correspond to the type of jobs usually submitted to them, or to the type of services they provide.

lost_and_found

If the LSF administrator removes queues from the system, LSF creates a queue called `lost_and_found` and places the jobs from the removed queues into the `lost_and_found` queue. Jobs in the `lost_and_found` queue will not be started unless they are switched to other queues (see `bswitch`).

PARAMETER/STATISTICS

PRIO

The priority of the queue. If job priority is not configured, determines the queue search order at job dispatch, suspension and resumption time. Queues with higher priority values are searched first for job dispatch and resumption (this is contrary to UNIX process priority ordering), and queues with higher priority values are searched last for job suspension.

STATUS

The current status of the queue. The possible values are:

Open

The queue is able to accept jobs.

Closed

The queue is not able to accept jobs.

Active

Jobs in the queue may be started.

Inactive

Jobs in the queue cannot be started for the time being.

At any moment, each queue is either `Open` or `Closed`, and is either `Active` or `Inactive`. The queue can be opened, closed, inactivated and re-activated by the LSF administrator using `badmin` (see `badmin(8)`). The queue becomes inactive when either its dispatch window is closed or its run window is closed (see DISPATCH_WINDOWS in the "Output for the -l Option" section). In this case, the queue cannot be activated using `badmin`. The queue is re-activated by LSF when one of its dispatch windows and one of its run windows are open again. The initial state of a queue at LSF boot time is set to open, and either active or inactive depending on its windows.

MAX

The maximum number of job slots that can be used by the jobs from the queue. These job slots are used by dispatched jobs which have not yet finished, and by pending jobs which have slots reserved for them. A sequential job will use one job slot

when it is dispatched to a host, while a parallel job will use as many job slots as is required by `bsub -n` when it is dispatched. See `bsub(1)` for details. If '–' is displayed, there is no limit.

JL/U

The maximum number of job slots you can use for your jobs in the queue. These job slots are used by your dispatched jobs which have not yet finished, and by pending jobs which have slots reserved for them. If '–' is displayed, there is no limit.

JL/P

The maximum number of job slots a processor can process from the queue. This includes job slots of dispatched jobs that have not yet finished, and job slots reserved for some pending jobs. The job slot limit per processor (JL/P) controls the number of jobs sent to each host. This limit is configured per processor so that multiprocessor hosts are automatically allowed to run more jobs. If '–' is displayed, there is no limit.

JL/H

The maximum number of job slots a host can process from the queue. This includes the job slots of dispatched jobs that have not yet finished, and those reserved for some pending jobs. The job slot limit per host (JL/H) controls the number of jobs sent to each host, regardless of whether a host is a uniprocessor host or a multiprocessor host. If '–' is displayed, there is no limit.

NJOBS

The total number of job slots held currently by jobs in the queue. This includes pending, running, suspended and reserved job slots. A parallel job that is running on *n* processors is counted as *n* job slots, since it takes *n* job slots in the queue. See `bjobs(1)` for an explanation of batch job states.

PEND

The number of pending job slots in the queue.

RUN

The number of running job slots in the queue.

SUSP

The number of suspended job slots in the queue.

Output for -l Option

In addition to the above fields, the -l option displays the following:

Description

A description of the typical use of the queue.

PARAMETERS/STATISTICS

NICE

The nice value at which jobs in the queue will be run. This is the UNIX nice value for reducing the process priority (see `nice(1)`).

STATUS

Inactive

The long format for the -l option gives the possible reasons for a queue to be inactive:

Inact_Win

The queue is out of its dispatch window or its run window.

Inact_Adm

The queue has been inactivated by the LSF administrator.

SSUSP

The number of job slots in the queue allocated to jobs that are suspended by LSF.

USUSP

The number of job slots in the queue allocated to jobs that are suspended by the job submitter or by the LSF administrator.

RSV

The numbers of job slots in the queue that are reserved by LSF for pending jobs.

Migration threshold

The length of time in seconds that a job dispatched from the queue will remain suspended by the system before LSF attempts to migrate the job to another host. See the MIG parameter in `lsb.queues` and `lsb.hosts`.

Schedule delay for a new job

The delay time in seconds for scheduling a session after a new job is submitted. If the schedule delay time is zero, a new scheduling session is started as soon as the job is submitted to the queue. See the NEW_JOB_SCHEDULE_DELAY parameter in `lsb.queues`.

Interval for a host to accept two jobs

The length of time in seconds to wait after dispatching a job to a host before dispatching a second job to the same host. If the job accept interval is zero, a host may accept more than one job in each dispatching interval. See the JOB_ACCEPT_INTERVAL parameter in `lsb.queues` and `lsb.params`.

RESOURCE LIMITS

The hard resource limits that are imposed on the jobs in the queue (see `getrlimit`(2) and `lsb.queues`(5)). These limits are imposed on a per-job and a per-process basis.

The possible per-job limits are:

CPULIMIT

PROCLIMIT

MEMLIMIT

SWAPLIMIT

PROCESSLIMIT

The possible UNIX per-process resource limits are:

RUNLIMIT

FILELIMIT

DATALIMIT

STACKLIMIT

CORELIMIT

If a job submitted to the queue has any of these limits specified (see `bsub(1)`), then the lower of the corresponding job limits and queue limits are used for the job.

If no resource limit is specified, the resource is assumed to be unlimited.

SCHEDULING PARAMETERS

The scheduling and suspending thresholds for the queue.

The scheduling threshold `loadSched` and the suspending threshold `loadStop` are used to control batch job dispatch, suspension, and resumption. The queue thresholds are used in combination with the thresholds defined for hosts (see `bhosts(1)` and `lsb.hosts`(5)). If both queue level and host level thresholds are configured, the most restrictive thresholds are applied.

The `loadSched` and `loadStop` thresholds have the following fields:

r15s

The 15-second exponentially averaged effective CPU run queue length.

r1m

The 1-minute exponentially averaged effective CPU run queue length.

r15m

The 15-minute exponentially averaged effective CPU run queue length.

ut

The CPU utilization exponentially averaged over the last minute, expressed as a percentage between 0 and 1.

pg

The memory paging rate exponentially averaged over the last minute, in pages per second.

io

The disk I/O rate exponentially averaged over the last minute, in kilobytes per second.

ls

The number of current login users.

it

On UNIX, the idle time of the host (keyboard not touched on all logged in sessions), in minutes.

On Windows NT, the `it` index is based on the time a screen saver has been active on a particular host.

tmp

The amount of free space in /tmp, in megabytes.

swp

The amount of currently available swap space, in megabytes.

mem

The amount of currently available memory, in megabytes.

In addition to these internal indices, external indices are also displayed if they are defined in `lsb.queues` (see `lsb.queues`(5)).

The `loadSched` threshold values specify the job dispatching thresholds for the corresponding load indices. If '–' is displayed as the value, it means the threshold is not applicable. Jobs in the queue may be dispatched to a host if the values of all the load indices of the host are within (below or above, depending on the meaning of the load index) the corresponding thresholds of the queue and the host. The same conditions are used to resume jobs dispatched from the queue that have been suspended on this host.

Similarly, the `loadStop` threshold values specify the thresholds for job suspension. If any of the load index values on a host go beyond the corresponding threshold of the queue, jobs in the queue will be suspended.

SCHEDULING POLICIES

Scheduling policies of the queue. Optionally, one or more of the following policies may be configured:

FAIRSHARE

Queue-level fairshare scheduling is enabled. Jobs in this queue are scheduled based on a fairshare policy instead of the first-come, first-serve (FCFS) policy.

BACKFILL

A job in a backfill queue can use the slots reserved by other jobs if the job can run to completion before the slot-reserving jobs start.

Backfilling does not occur on queue limits and user limit but only on host based limits. That is, backfilling is only supported when MXJ, JL/U, JL/P, PJOB_LIMIT, and HJOB_LIMIT are reached. Backfilling is not supported when MAX_JOBS, QJOB_LIMIT, and UJOB_LIMIT are reached.

IGNORE_DEADLINE

If IGNORE_DEADLINE is set to Y, starts all jobs regardless of the run limit.

EXCLUSIVE

Jobs dispatched from an exclusive queue can run exclusively on a host if the user so specifies at job submission time (see `bsub(1)`). Exclusive execution means that the job is sent to a host with no other batch job running there, and no further job, batch or interactive, will be dispatched to that host while the job is running. The default is not to allow exclusive jobs.

NO_INTERACTIVE

This queue does not accept batch interactive jobs. (see the `-I`, `-Is`, and `-Ip` options of `bsub(1)`). The default is to accept both interactive and non-interactive jobs.

ONLY_INTERACTIVE

This queue only accepts batch interactive jobs. Jobs must be submitted using the `-I`, `-Is`, and `-Ip` options of `bsub(1)`. The default is to accept both interactive and non-interactive jobs.

USER_SHARES

The user share assignment for a fairshare queue.

DEFAULT HOST SPECIFICATION

The default host or host model that will be used to normalize the CPU time limit of all jobs.

If you want to view a list of the CPU factors defined for the hosts in your cluster, use `lsinfo(1)`. The CPU factors are configured in `lsf.shared(5)`.

The appropriate CPU scaling factor of the host or host model is used to adjust the actual CPU time limit at the execution host (see CPULIMIT in `lsb.queues(5)`). The DEFAULT_HOST_SPEC parameter in `lsb.queues` overrides the system

DEFAULT_HOST_SPEC parameter in `lsb.params` (see `lsb.params(5)`). If a user explicitly gives a host specification when submitting a job using `bsub -c` *`cpu_limit`*`[/`*`host_name`* `|` `/`*`host_model`*`]`, the user specification overrides the values defined in both `lsb.params` and `lsb.queues`.

RUN_WINDOWS

One or more run windows in a week during which jobs in the queue may run.

When the end of a run window is reached, any running jobs from the queue are suspended until the beginning of the next run window when they are resumed. The default is no restriction, or always open.

DISPATCH_WINDOWS

The dispatch windows for the queue. The dispatch windows are the time windows in a week during which jobs in the queue may be dispatched.

When a queue is out of its dispatch window or windows, no job in the queue will be dispatched. Jobs already dispatched are not affected by the dispatch windows. The default is no restriction, or always open (that is, twenty-four hours a day, seven days a week). Note that such windows are only applicable to batch jobs. Interactive jobs scheduled by LIM are controlled by another set of dispatch windows (see `lshosts(1)`). Similar dispatch windows may be configured for individual hosts (see `bhosts(1)`).

A window is displayed in the format *begin_time–end_time*. Time is specified in the format [*day*:]*hour*[:*minute*], where all fields are numbers in their respective legal ranges: 0(Sunday)-6 for *day*, 0-23 for *hour*, and 0-59 for *minute*. The default value for *minute* is 0 (on the hour). The default value for *day* is every day of the week. The *begin_time* and *end_time* of a window are separated by '–', with no blank characters (SPACE and TAB) in between. Both *begin_time* and *end_time* must be present for a window. Windows are separated by blank characters.

USERS

A list of users and user groups allowed to submit jobs to the queue. User group names have a slash (/) added at the end of the group name. See `bugroup(1)`.

LSF cluster administrators can submit jobs to the queue by default (unless the fairshare scheduling policy is enabled).

If the fairshare scheduling policy is enabled, users cannot submit jobs to the queue unless they also have a share assignment.

HOSTS

A list of hosts and host groups where jobs in the queue can be dispatched. Host group names have a slash (/) added at the end of the group name. See `bmgroup(1)`.

NQS DESTINATION QUEUES

A list of NQS destination queues.

When you submit a job using `bsub -q` *queue_name*, and the specified queue is configured to forward jobs to the NQS system, LSF routes your job to one of the NQS destination queues. The job runs on an NQS batch server host, which is not a member of the LSF cluster. Although running on an NQS system outside the LSF cluster, the job is still managed by LSF in almost the same way as jobs running inside the LSF cluster. Thus, you may have your batch jobs transparently sent to an NQS system to run and then get the results of your jobs back. You may use any supported user interface, including LSF commands, `lsNQS` (the NQS commands; see `lsnqs(1)`), and `xlsbatch` (the GUI for Batch users; see `xlsbatch(1)`), to submit, monitor, signal and delete your batch jobs that are running in an NQS system. See `lsb.queues(5)` and `bsub(1)` for more information.

ADMINISTRATORS

A list of queue administrators. The users whose names are listed are allowed to operate on the jobs in the queue and on the queue itself. See `lsb.queues(5)` for more information.

PRE_EXEC

The queue's pre-execution command. The pre-execution command is executed before each job in the queue is run on the execution host (or on the first host selected for a parallel batch job). See `lsb.queues(5)` for more information.

POST_EXEC

The queue's post-execution command. The post-execution command is run when a job terminates. See `lsb.queues(5)` for more information.

REQUEUE_EXIT_VALUES

Jobs that exit with these values are automatically requeued. See `lsb.queues(5)` for more information.

RES_REQ

Resource requirements of the queue. Only the hosts that satisfy these resource requirements can be used by the queue.

Maximum slot reservation time

The maximum time in seconds a slot is reserved for a pending job in the queue. See the SLOT_RESERVE=MAX_RESERVE_TIME[n] parameter in `lsb.queues`.

RESUME_COND

Resume threshold conditions for a suspended job in the queue. See `lsb.queues(5)` for more information.

STOP_COND

Stop threshold conditions for a running job in the queue. See `lsb.queues(5)` for more information.

JOB_STARTER

Job starter command for a running job in the queue. See `lsb.queues(5)` for more information.

PREEMPTION

PREEMPTIVE

The queue is preemptive. Jobs in a preemptive queue may preempt running jobs from lower-priority queues, even if the lower-priority queues are not specified as preemptive.

PREEMPTABLE

The queue is preemptable. Running jobs in a preemptable queue may be preempted by jobs in higher-priority queues, even if the higher-priority queues are not specified as preemptive.

RERUNNABLE

If the RERUNNABLE field displays `yes`, jobs in the queue are rerunnable. That is, jobs in the queue are automatically restarted or rerun if the execution host becomes unavailable. However, a job in the queue will not be restarted if the you have removed the rerunnable option from the job. See `lsb.queues(5)` for more information.

CHECKPOINT

If the CHKPNTDIR field is displayed, jobs in the queue are checkpointable. Jobs will use the default checkpoint directory and period unless you specify other values. Note that a job in the queue will not be checkpointed if you have removed the checkpoint option from the job. See `lsb.queues(5)` for more information.

CHKPNTDIR

Specifies the checkpoint directory using an absolute or relative path name.

CHKPNTPERIOD

Specifies the checkpoint period in seconds.

Although the output of `bqueues` reports the checkpoint period in seconds, the checkpoint period is defined in minutes (the checkpoint period is defined through the `bsub -k "`*checkpoint_dir* [*checkpoint_period*]`"` option, or in `lsb.queues`).

JOB CONTROLS

The configured actions for job control. See JOB_CONTROLS parameter in `lsb.queues`.

The configured actions are displayed in the format [*action_type*, *command*] where *action_type* is either SUSPEND, RESUME, or TERMINATE.

Output for -r option

In addition to the fields displayed for the `-l` option, the `-r` option displays the following:

SCHEDULING POLICIES

FAIRSHARE

The `-r` option causes `bqueues` to recursively display the entire share information tree associated with the queue.

SEE ALSO

`lsfbatch(1), bugroup(1), nice(1), getrlimit(2), lsb.queues(5), bsub(1), bjobs(1), bhosts(1), badmin(8), mbatchd(8)`

bread

reads messages and attached data files from a job

SYNOPSIS

bread [**-i** *message_index*] [**-a** *file_name*]
job_ID | "*job_ID*[*index_list*]" | **-J** *job_name*

bread [**-h** | **-V**]

DESCRIPTION

Reads messages and data posted to a job with `bpost`.

By default, displays the message description text of the job. By default, operates on the first message (index 0) of the specified job.

Users can only read messages and data from their own jobs. Root and LSF administrators can also read messages of jobs submitted by other users; they cannot read data files attached to jobs owned by other users.

You can read messages and data from a job until it is cleaned from the system. You cannot read messages and data from done or exited jobs.

The command:

```
% bstatus
```

is equivalent to:

```
% bread -i 0
```

OPTIONS

`-i` *`message_index`*

Operates on the specified message index.

`-a` *`file_name`*

Gets the text message and copies the data file attached to the specified message index of the job to the specified file, overwriting the specified file if it already exists. To use `-a`, the job must have an attachment.

By default, `-a` gets the attachment file from the directory specified by the JOB_ATTA_DIR parameter. If JOB_ATTA_DIR is not specified, job message attachments are saved in `LSB_SHAREDIR/info/`.

job_ID | "*job_ID*[*index_list*]" | **-J** *job_name*

Required. Operates on the specified job.

-h

Prints command usage to stderr and exits.

-v

Prints LSF release version to stderr and exits.

EXAMPLE

```
% bpost -i 1 -d "step 1" -a step1.out 2500
% bread -i 1 -a step2.in 2500
JOBID      MSG_ID FROM      POST_TIME      DESCRIPTION
312        1      user1     May 19 13:59   step 1
```

Displays the message description text `step 1` for message index 1 of job 2500 and copies the data in the file `step1.out` attached to message 1 to the file `step2.in`.

SEE ALSO

bpost(1), bstatus(1), bsub(1), JOB_ATTA_DIR

brequeue

kills and requeues a job

SYNOPSIS

brequeue -J *job_name* | **-J** "*job_name*[*index_list*]" | **-u** *user_name* | **-u all** | *job_ID* | "*job_ID*[*index_list*]"

brequeue [**-h** | **-V**]

DESCRIPTION

You can only use this command on a job you own, unless you are `root` or an LSF administrator.

Kills a running (RUN), user-suspended (USUSP), or system-suspended (SSUSP) job and returns it to its original queue, with the same job ID. A job that is killed and requeued retains its submit time but is dispatched according to its requeue time. When the job is requeued, it is assigned the PEND status. Once dispatched, the job starts over from the beginning.

By default, kills your most recently submitted job.

OPTIONS

`-J` *`job_name`* **`|`** **`-J`** **`"`***`job_name`***`[`***`index_list`***`]"`**

Operates on the specified job.

For a job array, the index list, the square brackets, and the quotation marks are required.

Since job names are not unique, multiple job arrays may have the same name with a different or same set of indices.

`-u` *`user_name`* **`|`** **`-u all`**

Operates on the job submitted most recently by the specified user, or by all users if the keyword `all` is specified.

Only `root` and an LSF administrator can requeue jobs submitted by other users.

`job_ID` **`|"`***`job_ID`***`[`***`index_list`***`]"`**

Operates on the specified job or job array elements.

The value of 0 for *job_ID* is ignored.

-h

Prints command usage to stderr and exits.

-v

Prints LSF release version to stdout and exits.

LIMITATIONS

`brequeue` cannot be used on interactive batch jobs; `brequeue` only kills interactive batch jobs, it does not restart them.

brestart

restarts checkpointed jobs

SYNOPSIS

brestart [*bsub options*] [**-f**] *checkpoint_dir* [*job_ID* | "*job_ID***[***index***]**"]

brestart [**-h** | **-V**]

OPTION LIST

-B
-f
-N
-x
-b *begin_time*
-C *core_limit*
-c [*hour***:**]*minute*[*/host_name* | */host_model*]
-D *data_limit*
-E "*pre_exec_command* [*argument* ...]"
-F *file_limit*
-m "*host_name*[**+**[*pref_level*]] | *host_group*[**+**[*pref_level*]] ..."
-G *user_group*
-M *mem_limit*
-q "*queue_name* ..."
-S *stack_limit*
-t *term_time*
-w '*dependency_expression*'
-W *run_limit*[*/host_name* | */host_model*]
checkpoint_dir [*job_ID* | "*job_ID***[***index***]**"]
[**-h** | **-V**]

DESCRIPTION

Restarts a checkpointed job using the checkpoint files saved in *checkpoint_dir/last_job_ID/*. Only jobs that have been successfully checkpointed can be restarted.

Jobs are re-submitted and assigned a new job ID. The checkpoint directory is renamed using the new job ID, *checkpoint_dir/new_job_ID/*.

By default, jobs are restarted with the same output file and file transfer specifications, job name, window signal value, checkpoint directory and period, and rerun options as the original job.

To restart a job on another host, both hosts must be binary compatible, run the same OS version, have access to the executable, have access to all open files (LSF must locate them with an absolute path name), and have access to the checkpoint directory.

The environment variable LSB_RESTART is set to Y when a job is restarted.

LSF invokes the `erestart`(8) executable found in LSF_SERVERDIR to perform the restart.

Only the `bsub` options listed here can be used with `brestart`.

OPTIONS

Only the `bsub` options listed in the option list above can be used for `brestart`. Except for the following option, see `bsub`(`1`) for a description of `brestart` options.

`-f`

Forces the job to be restarted even if non-restartable conditions exist (these conditions are operating system specific).

SEE ALSO

`bsub(1), bjobs(1), bmod(1), bqueues(1), bhosts(1), bchkpnt(1), lsb.queues(5), echkpnt(8), erestart(8), mbatchd(8)`

LIMITATIONS

In kernel-level checkpointing, you cannot change the value of core limit, CPU limit, stack limit or memory limit with `brestart`.

bresume

resumes one or more suspended jobs

SYNOPSIS

bresume [**-R**] [**-J** *job_name*] [**-m** *host_name*] [**-q** *queue_name*] [**-u** *user_name* | **-u** *user_group* | **-u all**] [**0**]

bresume [**-R**] [*job_ID* | "*job_ID*[*index_list*]"] ...

bresume [**-h** | **-V**]

DESCRIPTION

Sends the SIGCONT signal to resume one or more of your suspended jobs. Only `root` and LSF administrators can operate on jobs submitted by other users. Using `bresume` on a job that is not in either the PSUSP or the USUSP state has no effect.

By default, resumes only one job, your most recently submitted job.

You can also use `bkill -s` to send the resume signal to a job.

You cannot resume a job that is not suspended.

OPTIONS

`-R`

Resumes jobs recursively on the job group tree.

`0`

Resumes all the jobs that satisfy other options (`-m`, `-q`, `-u` and `-J`).

`-J` *`job_name`*

Resumes only jobs with the specified name.

`-m` *`host_name`*

Resumes only jobs dispatched to the specified host.

`-q` *`queue_name`*

Resumes only jobs in the specified queue.

-u *user_name* | **-u** *user_group* | **-u all**

Resumes only jobs owned by the specified user or group, or all users is the reserved user name `all` is specified.

job_ID ... | "job_ID[*index_list*]*" ...*

Resumes only the specified jobs. Jobs submitted by any user can be specified here without using the `-u` option.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

EXAMPLES

```
% bresume -q night 0
```

Resumes all of the user's suspended jobs that are in the `night` queue. If the user is an LSF administrator, resumes all suspended jobs in the `night` queue.

SEE ALSO

`bsub(1), bjobs(1), bqueues(1), bhosts(1), bstop(1), bdel(1), bkill(1), bparams(5), mbatchd(8), kill(1), signal(2)`

brun

forces a job to run immediately

SYNOPSIS

brun [**-f**] **-m** "*host_name ...*" *job_ID*

brun [**-f**] **-m** "*host_name ...*" "*job_ID*[*index_list*]"

brun [**-h** | **-V**]

DESCRIPTION

This command can only be used by LSF administrators.

Forces a pending job to run immediately on specified hosts.

A job which has been forced to run is counted as a running job, this may violate the user, queue, or host job limits and fairshare priorities.

A job which has been forced to run cannot be preempted by other jobs even if it is submitted to a preemptable queue and other jobs are submitted to a preemptive queue.

By default, after the job is started, it is still subject to run windows and suspending conditions.

OPTIONS

-f

Allows the job to run without being suspended due to run windows or suspending conditions.

-m *host_name ...*

Required. Specify one or more hosts on which to run the job.

job_ID | "*job_ID*[*index_list*]"

Required. Specify the job to run, or specify one element of a job array.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

LIMITATIONS

You cannot force a job in SSUSP, USUSP or PSUSP state.

bstatus

gets current external job status or sets new job status

SYNOPSIS

bstatus [-d "*description*"**]** *job_ID* **|** "*job_ID***[***index_list***]"** **|** **-J** *job_name*

bstatus [-h | **-V]**

DESCRIPTION

Gets and displays the message description text of a job, or changes the contents of the message description text with the `-d` option.

By default, displays the message description text of the first message (index 0) of the specified job.

Users can only get or set external job status on their own jobs. Only root and LSF administrators can get or set external job status on jobs submitted by other users.

You can set the external status of a job until it completes. You cannot change the status of done or exited jobs. You can display the status of a job until it is cleaned from the system.

OPTIONS

-d "*description*"

Updates the job status with specified message description text.

-J *job_ID* | **-J** "*job_ID***[***index_list***]"** | **-J** *job_name*

Required. Operates on the specified job.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

EXAMPLES

```
% bstatus 2500
JOBID    FROM    UPDATE_TIME   STATUS
793      user1   Sep 14 16:54  step 1
```

Displays the message description text of the first message (index 0) of job 2500.

```
% bstatus -d "step 2" 2500
```

Changes the message description text of the first message (index 0) of job 2500 to `step 2`.

SEE ALSO

`bpost(1), bread(1)`

bstop

suspends unfinished jobs

SYNOPSIS

bstop [**-a**] [**-d**] [**-R**] [**0**] [**-J** *job_name*] [**-m** *host_name* | **-m** *host_group*] [**-q** *queue_name*] [**-u** *user_name* | **-u** *user_group* | **-u all**] [*job_ID* | "*job_ID***[***index***]**"] ...

bstop [**-h** | **-V**]

DESCRIPTION

Suspends unfinished jobs.

Sends the SIGSTOP signal to sequential jobs and the SIGTSTP signal to parallel jobs to suspend them.

By default, suspends only one job, your most recently submitted job.

Only `root` and LSF administrators can operate on jobs submitted by other users.

Using `bstop` on a job that is in the USUSP state has no effect.

You can also use `bkill -s` to send the suspend signal to a job, or send a resume signal to a job. You can use `bresume` to resume suspended jobs. You cannot suspend a job that is already suspended.

If a signal request fails to reach the job execution host, LSF will retry the operation later when the host becomes reachable. LSF retries the most recent signal request.

OPTIONS

-a

Suspends all jobs.

-d

Suspends only finished jobs (with a DONE or EXIT status).

-R

Suspends jobs recursively on the job group tree.

0

Suspends all the jobs that satisfy other options (`-m`, `-q`, `-u` and `-J`).

-J *job_name*

Suspends only jobs with the specified name.

-m *host_name* | **-m** *host_group*

Suspends only jobs dispatched to the specified host or host group.

-q *queue_name*

Suspends only jobs in the specified queue.

-u *user_name* | **-u** *user_group* | **-u all**

Suspends only jobs owned by the specified user or user group, or all users if the keyword `all` is specified.

job_ID ... | "*job_ID*[*index*]" ...

Suspends only the specified jobs. Jobs submitted by any user can be specified here without using the `-u` option.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

EXAMPLES

% bstop 314
Suspends job number 314.

% bstop -m apple
Suspends the invoker's last job that was dispatched to host `apple`.

% bstop -u smith 0
Suspends all the jobs submitted by user smith.

% bstop -u all
Suspends the last submitted job in the LSF system.

% bstop -u all 0
Suspends all the batch jobs in the LSF system.

SEE ALSO

bsub(1), bjobs(1), bqueues(1), bhosts(1), bresume(1), bdel(1), bkill(1), bparams(5), mbatchd(8), kill(1), signal(2)

bsub

submits a batch job to LSF

SYNOPSIS

bsub [*options*] *command* [*arguments*]
bsub [**-h** | **-V**]

OPTION LIST

-B
-H
[**-I** | **-Ip** | **-Is** | **-K**]
-N
-r
-x
-b *begin_time*
-C *core_limit*
-c [*hour***:**]*minute*[*/host_name* | */host_model*]
-D *data_limit*
-e *err_file*
-extsched "**CPU_LIST=***cpu_ID_list***;CPUSET_OPTIONS=***option_list*"
-E "*pre_exec_command* [*argument* ...]"
-f "*local_file op* [*remote_file*]" ...
-F *file_limit*
-G *user_group*
[**-i** *input_file* | **-is** *input_file*]
-J *job_name* | **-J** "*job_name***[***index_list***]%***job_limit*"
-k "*checkpoint_dir* [*checkpoint_period*] [**method=***method_name*]"
-L *login_shell*
-m "*host_name*[**+**[*pref_level*]] | *host_group*[**+**[*pref_level*]] ..."
-M *mem_limit*
-n *min_processors*[**,***max_processors*]
-o *out_file*
-P *project_name*
-q "*queue_name* ..."
-R "*res_req*"
-**sp** *priority*
-S *stack_limit*
-t *term_time*

-u *mail_user*
-w '*dependency_expression*'
-W *run_limit*[*/host_name* | */host_model*]
-Zs

DESCRIPTION

Submits a job for batch execution and assigns it a unique numerical job ID.

Runs the job on a host that satisfies all requirements of the job, when all conditions on the job, host, queue, and cluster are satisfied. If LSF cannot run all jobs immediately, LSF scheduling policies determine the order of dispatch. Jobs are started and suspended according to the current system load.

Sets the user's execution environment for the job, including the current working directory, file creation mask, and all environment variables, and sets LSF environment variables before starting the job.

When a job is run, the command line and stdout/stderr buffers are stored in the directory *home_directory*`/.lsbatch` on the execution host. If this directory is not accessible, `/tmp/.lsbtmp`*user_ID* is used as the job's home directory. If the current working directory is under the home directory on the submission host, then the current working directory is also set to be the same relative directory under the home directory on the execution host. The job is run in `/tmp` if the current working directory is not accessible on the execution host.

If no command is supplied for `bsub`, `bsub` prompts for the command from the standard input. On UNIX, the input is terminated by entering CTRL-D on a new line. On Windows NT, the input is terminated by entering CTRL-Z on a new line.

By default, LSF assumes that uniform user names and user ID spaces exist among all the hosts in the cluster. That is, a job submitted by a given user will run under the same user's account on the execution host. For situations where nonuniform user names and user ID spaces exist, account mapping must be used to determine the account used to run a job.

By default, uses the command name as the job name. Quotation marks are significant.

By default, if fairshare is defined and you belong to multiple user groups, the job will be scheduled under the user group that allows the quickest dispatch.

By default, the job is not checkpointable.

By default, automatically selects an appropriate queue. If you defined a default queue list by setting LSB_DEFAULTQUEUE, the queue is selected from your list. If LSB_DEFAULTQUEUE is not defined, the queue is selected from the system default queue list specified by the LSF administrator (see the parameter DEFAULT_QUEUE in `lsb.params`(5)).

By default, LSF tries to obtain resource requirement information for the job from the remote task list that is maintained by the load sharing library (see `lsfintro`(1)). If the job is not listed in the remote task list, the default resource requirement is to run the job on a host or hosts that are of the same host type (see `lshosts`(1)) as the submission host.

By default, assumes only one processor is requested.

By default, does not start a login shell but runs the job file under the execution environment from which the job was submitted.

By default, the input file for the batch job is `/dev/null` (no input).

By default, sends mail to you when the job is done. The default destination is defined by LSB_MAILTO in `lsf.conf`. The mail message includes the job report, the job output (if any), and the error message (if any). By default, charges the job to the default project. The default project is the project you define by setting the environment variable LSB_DEFAULTPROJECT. If you do not set LSB_DEFAULTPROJECT, the default project is the project specified by the LSF administrator in the `lsb.params` configuration file (see the DEFAULT_PROJECT parameter in `lsb.params`(5)). If DEFAULT_PROJECT is not defined, then LSF uses `default` as the default project name.

Use `-n` to submit a parallel job.

Use `-I`, `-Is`, or `-Ip` to submit a batch interactive job.

Use `-J` to assign a name to your job.

Use `-k` to specify a checkpointable job.

To kill a batch job submitted with `bsub`, use `bkill`.

Jobs submitted to a chunk job queue with the following options are not chunked; they are dispatched individually:

-k (checkpointable jobs)

-r (rerunnable jobs)

-I (interactive jobs)

-c (jobs with CPU limit greater than 30)

-W (jobs with run limit greater than 30 minutes)

Use bmod to modify jobs submitted with bsub. bmod takes similar options to bsub.

OPTIONS

-B

Sends mail to you when the job is dispatched and begins execution.

-H

Holds the job in the PSUSP state when the job is submitted. The job will not be scheduled until you tell the system to resume the job (see bresume(1)).

-I | -Ip | -Is

Submits a batch interactive job. A new job cannot be submitted until the interactive job is completed or terminated.

Sends the job's standard output (or standard error) to the terminal. Does not send mail to you when the job is done unless you specify the -N option.

Terminal support is available for a batch interactive job.

When you specify the -Ip option, submits a batch interactive job and creates a pseudo-terminal when the job starts. Some applications (for example, vi) require a pseudo-terminal in order to run correctly.

When you specify the -Is option, submits a batch interactive job and creates a pseudo-terminal with shell mode support when the job starts. This option should be specified for submitting interactive shells, or applications which redefine the CTRL-C and CTRL-Z keys (for example, jove).

If the -i *input_file* option is specified, you cannot interact with the job's standard input via the terminal.

If the `-o` *out_file* option is specified, sends the job's standard output to the specified output file. If the `-e` *err_file* option is specified, sends the job's standard error to the specified error file.

You cannot use `-I`, `-Ip`, or `-Is` with the `-K` option.

`-K`

Submits a batch job and waits for the job to complete. Sends the message "`Waiting for dispatch`" to the terminal when you submit the job. Sends the message `"Job is finished"` to the terminal when the job is done.

You will not be able to submit another job until the job is completed. This is useful when completion of the job is required in order to proceed, such as a job script. If the job needs to be rerun due to transient failures, `bsub` returns after the job finishes successfully. `bsub` will exit with the same exit code as the job so that job scripts can take appropriate actions based on the exit codes. `bsub` exits with value 126 if the job was terminated while pending.

You cannot use the `-K` option with the `-I`, `-Ip`, or `-Is` options.

`-N`

Sends the job report to you by mail when the job finishes. When used without any other options, behaves the same as the default.

Use only with `-o`, `-I`, `-Ip`, and `-Is` options, which do not send mail, to force LSF to send you a mail message when the job is done.

`-r`

If the execution host becomes unavailable while a job is running, specifies that the job will rerun on another host. LSF requeues the job in the same job queue with the same job ID. When an available execution host is found, reruns the job as if it were submitted new. You receive a mail message informing you of the host failure and requeuing of the job.

If the system goes down while a job is running, specifies that the job will be requeued when the system restarts.

Reruns a job if the execution host or the system fails; it does not rerun a job if the job itself fails.

If the execution host becomes unavailable after a job has been checkpointed (see **bsub -k** and **bchkpnt**(1)), the job is restarted from the last checkpoint. The restarted job is requeued for execution in the same way that you would restart a job using **brestart**(1). In order for the job to be successfully restarted, the job's checkpoint directory must reside in a shared file system accessible to the host receiving the restarted job.

-x

Puts the host running your job into exclusive execution mode.

In exclusive execution mode, your job runs by itself on a host. It is dispatched only to a host with no other jobs running, and LSF does not send any other jobs to the host until the job completes.

To submit a job in exclusive execution mode, the queue must be configured to allow exclusive jobs.

When the job is dispatched, **bhosts**(1) reports the host status as **closed_Excl**, and **lsload**(1) reports the host status as **lockU**.

Until your job is complete, the host is not selected by LIM in response to placement requests made by **lsplace**(1), **lsrun**(1) or **lsgrun**(1) or any other load sharing applications.

You can force other batch jobs to run on the host by using the **-m** *host_name* option of **brun**(1) to explicitly specify the locked host.

You can force LIM to run other interactive jobs on the host by using the **-m** *host_name* option of **lsrun**(1) or **lsgrun**(1) to explicitly specify the locked host.

-b *begin_time*

Dispatches the job for execution on or after the specified date and time. The date and time are in the form of [[*month*:]*day*:]*hour*:*minute* where the number ranges are as follows: month 1-12, day 1-31, hour 0-23, minute 0-59.

At least two fields must be specified. These fields are assumed to be *hour*:*minute*. If three fields are given, they are assumed to be *day*:*hour*:*minute*, and four fields are assumed to be *month*:*day*:*hour*:*minute*.

-C *core_limit*

Sets a per-process (soft) core file size limit for all the processes that belong to this batch job (see `getrlimit`(2)). The core limit is specified in kilobytes.

The behavior of this option depends on platform-specific UNIX systems.

In some cases, the process is sent a SIGXFSZ signal if the job attempts to create a core file larger than the specified limit. The SIGXFSZ signal normally terminates the process.

In other cases, the writing of the core file terminates at the specified limit.

-c [*hour*:]*minute*[*/host_name* | */host_model*]

Limits the total CPU time the job can use. This option is useful for preventing runaway jobs or jobs that use up too many resources. When the total CPU time for the whole job has reached the limit, a SIGXCPU signal is first sent to the job, then SIGINT, SIGTERM, and SIGKILL.

If LSB_JOB_CPULIMIT in `lsf.conf` is set to n, LSF-enforced CPU limit is disabled and LSF passes the limit to the operating system. When one process in the job exceeds the CPU limit, the limit is enforced by the operating system.

The CPU limit is in the form of [*hour*:]*minute*. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

Optionally, you can supply a host name or a host model name defined in LSF. You must insert '/' between the CPU limit and the host name or model name. (See `lsinfo`(1) to get host model information.) If a host name or model name is not given, LSF uses the default CPU time normalization host defined at the queue level (DEFAULT_HOST_SPEC in `lsb.queues`) if it has been configured, otherwise uses the default CPU time normalization host defined at the cluster level (DEFAULT_HOST_SPEC in `lsb.params`) if it has been configured, otherwise uses the submission host.

The CPU time you specify is the normalized CPU time. This is done so that the job does approximately the same amount of processing for a given CPU limit, even if it is sent to host with a faster or slower CPU. Whenever a normalized CPU time is given, the actual time on the

execution host is the specified time multiplied by the CPU factor of the normalization host then divided by the CPU factor of the execution host.

-D *data_limit*

Sets a per-process (soft) data segment size limit for each of the processes that belong to the batch job (see `getrlimit`(2)). The data limit is specified in kilobytes. A `sbrk` call to extend the data segment beyond the data limit will return an error.

-e *err_file*

Specify a file path. Appends the standard error output of the job to the specified file.

If you use the special character `%J` in the name of the error file, then `%J` is replaced by the job ID of the job. If you use the special character `%I` in the name of the error file, then `%I` is replaced by the index of the job in the array if the job is a member of an array. Otherwise, `%I` is replaced by 0 (zero).

If the current working directory is not accessible on the execution host after the job starts, LSF writes the standard error output file to `/tmp/`.

-E "*pre_exec_command* [*arguments ...*]"

Runs the specified pre-exec command on the batch job's execution host before actually running the job. For a parallel job, the pre-exec command runs on the first host selected for the parallel job. If the pre-exec command exits with 0, then the real job is started on the selected host. Otherwise, the job (including the pre-exec command) goes back to PEND status and is rescheduled.

If your job goes back into PEND status, LSF will keep on trying to run the pre-exec command and the real job when conditions permit. For this reason, be sure that your pre-exec command can be run many times without having side effects.

The standard input and output for the pre-exec command are directed to the same files as for the real job. The pre-exec command runs under the same user ID, environment, home, and working directory as the real job. If the pre-exec command is not in the user's normal execution path (the $PATH variable), the full path name of the command must be specified.

-extsched **-extsched "CPU_LIST=***cpu_ID_list***;CPUSET_OPTIONS=***option_list***"**

Specifies a list of CPUs and cpuset attributes used by LSF to allocate an SGI IRIX cpuset.

cpu_ID_list

is a list of CPU IDs separated by commas. The CPU ID is a positive integer or a range of integers. If incorrect CPU IDs are specified, the job is rejected.

option_list

is a list of cpuset attributes separated by commas. If incorrect cpuset attributes are specified, the job is rejected.

When a job is submitted using `-extsched`, LSF creates an IRIX cpuset with the specified CPUs and cpuset attributes and attaches it to the processes of the job. The job is then scheduled and dispatched.

If LSF_DEFAULT_EXTSCHED is set in `lsf.conf`, and the job is submitted without `-extsched` options, the options specified in LSF_DEFAULT_EXTSCHED are used to allocate the cpuset.

The following cpuset attributes are supported in the list of cpuset options specified by CPUSET_OPTIONS:

- CPUSET_CPU_EXCLUSIVE—defines a restricted cpuset
- CPUSET_MEMORY_LOCAL—threads assigned to the cpuset attempt to assign memory only from nodes within the cpuset
- CPUSET_MEMORY_EXCLUSIVE—threads assigned to the cpuset attempt to assign memory only from nodes within the cpuset
- CPUSET_MEMORY_KERNEL_AVOID—kernel attempts to avoid allocating memory from nodes contained in this cpuset
- CPUSET_MEMORY_MANDATORY—kernel limits all memory allocations to nodes contained in this cpuset
- CPUSET_POLICY_PAGE—Causes the kernel to page user pages to the swap file to free physical memory on the nodes contained in this cpuset. This is the default policy if no other policy is specified. Requires CPUSET_MEMORY_MANDATORY.
- CPUSET_POLICY_KILL—The kernel attempts to free as much space as possible from kernel heaps, but will not page user pages to the swap file. Requires CPUSET_MEMORY_MANDATORY.

See the IRIX 6.5.8 resource administration documentation and the man pages for the IRIX `cpuset` command for information about cpuset attributes.

`-f "`*`local_file op [remote_file]`***`" ...`**

Copies a file between the local (submission) host and the remote (execution) host. Specify absolute or relative paths, including the file names. You should specify the remote file as a file name with no path when running in non-shared systems.

If the remote file is not specified, it defaults to the local file, which must be given. Use multiple `-f` options to specify multiple files.

op

An operator that specifies whether the file is copied to the remote host, or whether it is copied back from the remote host. The operator must be surrounded by white space.

The following describes the operators:

`>` Copies the local file to the remote file before the job starts. Overwrites the remote file if it exists.

`<` Copies the remote file to the local file after the job completes. Overwrites the local file if it exists.

`<<` Appends the remote file to the local file after the job completes. The local file must exist.

`><` Copies the local file to the remote file before the job starts. Overwrites the remote file if it exists. Then copies the remote file to the local file after the job completes. Overwrites the local file.

`<>` Copies the local file to the remote file before the job starts. Overwrites the remote file if it exists. Then copies the remote file to the local file after the job completes. Overwrites the local file.

If you use the `-i` *input_file* option, then you do not have to use the `-f` option to copy the specified input file to the execution host. LSF does this for you, and removes the input file from the execution host after the job completes.

If you use the `-e` *err_file* or the `-o` *out_file* option, and you want the specified file to be copied back to the submission host when the job completes, then you must use the `-f` option.

If the submission and execution hosts have different directory structures, you must ensure that the directory where the remote file and local file will be placed exists.

If the local and remote hosts have different file name spaces, you must always specify relative path names. If the local and remote hosts do not share the same file system, you must ensure that the directory containing the remote file exists. It is recommended that only the file name be given for the remote file when running in heterogeneous file systems. This places the file in the job's current working directory. If the file is shared between the submission and execution hosts, then no file copy is performed.

LSF uses `lsrcp` to transfer files (see `lsrcp`(1) command). `lsrcp` contacts RES on the remote host to perform the file transfer. If RES is not available, `rcp` is used (see `rcp`(1)). The user must ensure that the `rcp` binary is in the user's $PATH on the execution host.

Jobs that are submitted from LSF client hosts should specify the `-f` option only if `rcp` is allowed. Similarly, `rcp` must be allowed if account mapping is used.

`-F` *`file_limit`*

Sets a per-process (soft) file size limit for each of the processes that belong to the batch job (see `getrlimit`(2)). The file size limit is specified in kilobytes. If a job process attempts to write to a file that exceeds the file size limit, then that process is sent a SIGXFSZ signal. The SIGXFSZ signal normally terminates the process.

`-G` *`user_group`*

Only useful with fairshare scheduling.

Associates the job with the specified group. Specify any group that you belong to that does not contain any subgroups. You must be a direct member of the specified user group.

-i *input_file* | **-is** *input_file*

Gets the standard input for the job from specified file. Specify an absolute or relative path. The input file can be any type of file, though it is typically a shell script text file.

If the file exists on the execution host, LSF uses it. Otherwise, LSF attempts to copy the file from the submission host to the execution host. For the file copy to be successful, you must allow remote copy (`rcp`) access, or you must submit the job from a server host where RES is running. The file is copied from the submission host to a temporary file in the directory specified by the JOB_SPOOL_DIR parameter, or your `$HOME/.lsbatch` directory on the execution host. LSF removes this file when the job completes.

The `-is` option spools the input file to the directory specified by the JOB_SPOOL_DIR parameter in `lsb.params`, and uses the spooled file as the input file for the job. By default, if JOB_SPOOL_DIR is not specified, the input file is spooled to `LSB_SHAREDIR/`*cluster_name*`/lsf_indir`. If the `lsf_indir` directory does not exist, LSF creates it before spooling the file. LSF removes the spooled file when the job completes. Use the `-is` option if you need to modify or remove the input file before the job completes. Removing or modifying the original input file does not affect the submitted job.

Unless you use `-is`, you can use the special characters `%J` and `%I` in the name of the input file. `%J` is replaced by the job ID. `%I` is replaced by the index of the job in the array, if the job is a member of an array, otherwise by 0 (zero). The special characters `%J` and `%I` are not valid with the `-is` option.

-J *job_name*
-J "*job_name*[*index_list*]%*job_slot_limit*"

Assigns the specified name to the job, and, for job arrays, specifies the indices of the job array and optionally the maximum number of jobs that can run at any given time.

The job name need not be unique.

To specify a job array, enclose the index list in square brackets, as shown, and enclose the entire job array specification in quotation marks, as shown. The index list is a comma-separated list whose elements have the syntax *start*[-*end*[**:***step*]] where *start*, *end* and *step* are

positive integers. If the step is omitted, a step of one is assumed. The job array index starts at one. By default, the maximum job array index is 1000.

You may also use a positive integer to specify the system-wide job slot limit (the maximum number of jobs that can run at any given time) for this job array.

All jobs in the array share the same job ID and parameters. Each element of the array is distinguished by its array index.

After a job is submitted, you use the job name to identify the job. Specify "*job_ID***[***index***]**" to work with elements of a particular array. Specify "*job_name***[***index***]**" to work with elements of all arrays with the same name. Since job names are not unique, multiple job arrays may have the same name with a different or same set of indices.

-k "*checkpoint_dir* **[***checkpoint_period***][method=***method_name***]"**

Makes a job checkpointable and specifies the checkpoint directory. If you omit the checkpoint period, the quotes are not required. Specify a relative or absolute path name.

When a job is checkpointed, the checkpoint information is stored in *checkpoint_dir/job_ID/file_name*. Multiple jobs can checkpoint into the same directory. The system can create multiple files.

The checkpoint directory is used for restarting the job (see `brestart`(1)).

Optionally, specifies a checkpoint period in minutes. Specify a positive integer. The running job is checkpointed automatically every checkpoint period. The checkpoint period can be changed using `bchkpnt`(1). Because checkpointing is a heavyweight operation, you should choose a checkpoint period greater than half an hour.

Optionally, specifies a custom checkpoint and restart method to use with the job. Use **`method=default`** to indicate to use LSF's default checkpoint and restart programs for the job, `echkpnt.default` and `erestart.default`.

The `echkpnt.`*`method_name`* and `erestart.`*`method_name`* programs must be in LSF_SERVERDIR or in the directory specified by LSB_ECHKPNT_METHOD_DIR (environment variable or set in `lsf.conf`).

If a custom checkpoint and restart method is already specified with LSB_ECHKPNT_METHOD (environment variable or in `lsf.conf`), the method you specify with `bsub -k` overrides this.

Process checkpointing is not available on all host types, and may require linking programs with a special libraries (see `libckpt.a(3)`). LSF invokes `echkpnt` (see `echkpnt(8)`) found in LSF_SERVERDIR to checkpoint the job. You can override the default `echkpnt` for the job by defining as environment variables or in `lsf.conf` LSB_ECHKPNT_METHOD and LSB_ECHKPNT_METHOD_DIR to point to your own `echkpnt`. This allows you to use other checkpointing facilities, including application-level checkpointing.

`-L` *`login_shell`*

Initializes the execution environment using the specified login shell. The specified login shell must be an absolute path. This is not necessarily the shell under which the job will be executed.

`-m "`*`host_name`***`[+[`***`pref_level`***`]] | `***`host_group`***`[+[`***`pref_level`***`]] ..."`**

Runs the job on one of the specified hosts.

By default, if multiple hosts are candidates, runs the job on the least-loaded host. To change this, put a plus (+) after the names of hosts or host groups that you would prefer to use, optionally followed by a preference level. For preference level, specify a positive integer, with higher numbers indicating greater preferences for those hosts.

For example, `-m "hostA groupB+2 hostC+1"` indicates that `groupB` is the most preferred and `hostA` is the least preferred.

For information about host groups, use `bmgroup`.

The keyword `others` can be specified with or without a preference level to refer to other hosts not otherwise listed. The keyword `others` must be specified with at least one host name or host group, it cannot be specified by itself. For example, `-m "hostA+ others"` means that `hostA` is preferred over all other hosts.

If you use both the `-m "`*`host_name`*`[+[`*`pref_level`*`]] | `*`host_group`*`[+[`*`pref_level`*`]]..."` option and the `-q` *queue_name* option, the specified queue must be configured to include all the hosts in the your host list. Otherwise, the job is not submitted. To find out what hosts are configured for the queue, use `bqueues -l`.

-M *mem_limit*

Specify the memory limit, in kilobytes.

If LSB_MEMLIMIT_ENFORCE or LSB_JOB_MEMLIMIT are set to `y` in `lsf.conf`, LSF kills the job when it exceeds the memory limit. Otherwise, LSF passes the memory limit to the operating system. UNIX operating systems that support RUSAGE_RSS for `setrlimit()` can apply the memory limit to each process. The following operating systems do not support the memory limit at the OS level:

- Windows NT

- Sun Solaris 2.x

-n *min_proc*[,*max_proc*]

Submits a parallel job and specifies the minimum and maximum numbers of processors required to run the job (some of the processors may be on the same multiprocessor host). If you do not specify a maximum, the number you specify represents the exact number of processors to use.

If the maximum number of processors is greater than the process limit of the queue to which the job is submitted, LSF will reject the job (see the PROCLIMIT parameter in `lsb.queues`(5)).

Once at least the minimum number of processors is available, the job is dispatched to the first host selected. The list of selected host names for the job are specified in the environment variables LSB_HOSTS and LSB_MCPU_HOSTS. The job itself is expected to start parallel components on these hosts and establish communication among them, optionally using RES.

-o *out_file*

Specify a file path. Appends the standard output of the job to the specified file. Sends the output by mail if the file does not exist, or the system has trouble writing to it.

If the current working directory is not accessible on the execution host after the job starts, LSF writes the standard output file to `/tmp/`.

If only a file name is specified, LSF writes the output file to the

If you use `-o` without `-e`, the standard error of the job is stored in the output file.

If you use `-o` without `-N`, the job report is stored in the output file as the file header.

If you use both `-o` and `-N`, the output is stored in the output file and the job report is sent by mail. The job report itself does not contain the output, but the report will advise you where to find your output.

If you use the special character `%J` in the name of the output file, then `%J` is replaced by the job ID of the job. If you use the special character `%I` in the name of the output file, then `%I` is replaced by the index of the job in the array, if the job is a member of an array. Otherwise, %I is replaced by 0 (zero).

`-P` *`project_name`*

Assigns the job to the specified project.

On IRIX 6, you must be a member of the project as listed in `/etc/project`(4). If you are a member of the project, then `/etc/projid`(4) maps the project name to a numeric project ID. Before the submitted job executes, a new array session (`newarraysess`(2)) is created and the project ID is assigned to it using `setprid`(2).

`-q "`*`queue_name ...`*`"`

Submits the job to one of the specified queues. Quotes are optional for a single queue. For a list of available queues, use `bqueues`.

When a list of queue names is specified, LSF selects the most appropriate queue in the list for your job based on the job's resource limits, and other restrictions, such as the requested hosts, your accessibility to a queue, queue status (closed or open), whether a queue can accept exclusive jobs, etc. The order in which the queues are considered is the same order in which these queues are listed. The queue listed first is considered first.

`-R "`*`res_req`*`"`

Runs the job on a host that meets the specified resource requirements. Specify the resource requirement string as usual. The size of the resource requirement string is limited to 512 bytes.

Any run-queue-length-specific resource, such as `r15s`, `r1m` or `r15m`, specified in the resource requirements refers to the normalized run queue length.

-sp *priority*

Specifies user-assigned job priority which allow users to order their jobs in a queue. Valid values for priority are any integers between 1 and MAX_USER_PRIORITY (displayed by `bparams -l`). Invalid job priorities are rejected. LSF and queue administrators can specify priorities beyond MAX_USER_PRIORITY.

The job owner can change the priority of their own jobs. LSF and queue administrators can change the priority of all jobs in a queue.

Job order is the first consideration to determine job eligibility for dispatch. Jobs are still subject to all scheduling policies regardless of job priority. Jobs with the same priority are ordered first come first served.

User-assigned job priority can be configured with automatic job priority escalation to automatically increase the priority of jobs that have been pending for a specified period of time.

-S *stack_limit*

Sets a per-process (soft) stack segment size limit (KB) for each of the processes that belong to the batch job (see `getrlimit`(2)).

-t *term_time*

Specifies the job termination deadline. If a UNIX job is still running at the termination time, the job is sent a SIGUSR2 signal, and is killed if it does not terminate within ten minutes. If a Windows NT job is still running at the termination time, it is killed immediately. (For a detailed description of how these jobs are killed, see `bkill`.) In the queue definition, a TERMINATE action can be configured to override the `bkill` default action (see the JOB_CONTROLS parameter in `lsb.queues`(5)).

The format for the termination time is [[*month*:]*day*:]*hour*:*minute* where the number ranges are as follows: month 1-12, day 1-31, hour 0-23, minute 0-59.

At least two fields must be specified. These fields are assumed to be *hour*:*minute*. If three fields are given, they are assumed to be *day*:*hour*:*minute*, and four fields are assumed to be *month*:*day*:*hour*:*minute*.

-u *mail_user*

Sends mail to the specified email destination.

-w *'dependency_expression'*

LSF will not place your job unless the dependency expression evaluates to TRUE. If you specify a dependency on a job that LSF cannot find (such as a job that has not yet been submitted), your job submission fails.

The dependency expression is a logical expression composed of one or more dependency conditions. To make dependency expression of multiple conditions, use the following logical operators:

&& (AND)

|| (OR)

! (NOT)

Use parentheses to indicate the order of operations, if necessary.

Enclose the dependency expression in single quotes (') to prevent the shell from interpreting special characters (space, any logic operator, or parentheses). If you use single quotes for the dependency expression, use double quotes for quoted items within it, such as job names.

In dependency conditions, job names specify only your own jobs, unless you are an LSF administrator. By default, if you use the job name to specify a dependency condition, and more than one of your jobs has the same name, all of your jobs that have that name must satisfy the test. If JOB_DEP_LAST_SUB in `lsb.params` is set to 1, the test is done on the job submitted most recently. Use double quotes (") around job names that begin with a number. In the job name, specify the wildcard character asterisk (*) at the end of a string, to indicate all jobs whose name begins with the string. For example, if you use `jobA*` as the job name, it specifies jobs named `jobA`, `jobA1`, `jobA_test`, `jobA.log`, etc.

In dependency conditions, the variable *op* represents one of the following relational operators:

>

>=

<

<=

==

!=

Use the following conditions to form the dependency expression.

done(*job_ID |"job_name" ...***)**

The job state is DONE.

ended(*job_ID | "job_name"***)**

The job state is EXIT or DONE.

exit(*job_ID | "job_name"* [,[*op*] *exit_code*]**)**

The job state is EXIT, and the job's exit code satisfies the comparison test.

If you specify an exit code with no operator, the test is for equality (== is assumed).

If you specify only the job, any exit code satisfies the test.

external(*job_ID | "job_name", "status_text"***)**

Specify the first word of the job status or message description (no spaces). Only the first word is evaluated.

The job has the specified job status, or the text of the job's status begins with the specified word.

job_ID | "job_name"

If you specify a job without a dependency condition, the test is for the DONE state (LSF assumes the "done" dependency condition by default).

numdone(*job_ID, op number | ***)**

For a job array, the number of jobs in the DONE state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numended(*job_ID, op number | ***)**

For a job array, the number of jobs in the DONE or EXIT states satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numexit(*job_ID, op number | ***)**

For a job array, the number of jobs in the EXIT state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numhold(*job_ID, op number | ***)**

For a job array, the number of jobs in the PSUSP state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numpend(*job_ID, op number | ***)**

For a job array, the number of jobs in the PEND state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numrun(*job_ID, op number | ***)**

For a job array, the number of jobs in the RUN state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numstart(*job_ID, op number | ***)**

For a job array, the number of jobs in the RUN, USUSP, or SSUSP states satisfies the test. Use * (with no operator) to specify all the jobs in the array.

post_done(*job_ID | "job_name"***)**

The job state is POST_DONE (the post-processing of specified job has completed without errors).

post_err(*job_ID | "job_name"***)**

The job state is POST_ERR (the post-processing of the specified job has completed with errors).

started(*job_ID | "job_name"***)**

The job state is:

- RUN, DONE, or EXIT

- PEND or PSUSP, and the job has a pre-execution command (`bsub -E`) that is running.

-W [*hour*:]*minute*[*/host_name* | */host_model*]

Sets the run time limit of the batch job. If a UNIX job runs longer than the specified run limit, the job is sent a SIGUSR2 signal, and is killed if it does not terminate within ten minutes. If a Windows NT job runs longer than the specified run limit, it is killed immediately. (For a detailed description of how these jobs are killed, see **bkill**.) In the queue definition, a TERMINATE action can be configured to override the **bkill** default action (see the JOB_CONTROLS parameter in **lsb.queues**(5)).

The run limit is in the form of [*hour***:**]*minute*. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

Optionally, you can supply a host name or a host model name defined in LSF. You must insert '**/**' between the CPU limit and the host name or model name. (See **lsinfo**(1) to get host model information.) If a host name or model name is not given, LSF uses the default CPU time normalization host defined at the queue level (DEFAULT_HOST_SPEC in **lsb.queues**) if it has been configured, otherwise uses the default CPU time normalization host defined at the cluster level (DEFAULT_HOST_SPEC in **lsb.params**) if it has been configured, otherwise uses the submission host.

The CPU time you specify is the normalized CPU time. This is done so that the job does approximately the same amount of processing, even if it is sent to host with a faster or slower CPU. Whenever a normalized CPU time is given, the actual time on the execution host is the specified time multiplied by the CPU factor of the normalization host then divided by the CPU factor of the execution host.

If the job also has termination time specified through the **bsub -t** *term_time* option, LSF determines whether the job can actually run for the specified length of time allowed by the run limit before the termination time. If not, then the job will be aborted. If the IGNORE_DEADLINE parameter is set in **lsb.queues**(5), this behavior is overridden and the run limit is ignored.

-Zs

Spools a job command file to the directory specified by the JOB_SPOOL_DIR parameter in **lsb.params**, and uses the spooled file as the command file for the job.

By default, if JOB_SPOOL_DIR is not specified, the input file is spooled to `LSB_SHAREDIR/cluster_name/lsf_cmddir`. If the `lsf_cmddir` directory does not exist, LSF creates it before spooling the file. LSF removes the spooled file when the job completes.

The `-Zs` option is not supported for embedded job commands because LSF is unable to determine the first command to be spooled in an embedded job command.

`-h`

Prints command usage to stderr and exits.

`-V`

Prints LSF release version to stderr and exits.

`command [argument]`

The job can be specified by a command line argument *command*, or through the standard input if the command is not present on the command line. The *command* can be anything that is provided to a UNIX Bourne shell (see `sh(1)`). *command* is assumed to begin with the first word that is not part of a `bsub` option. All arguments that follow *command* are provided as the arguments to the *command*.

If the batch job is not given on the command line, `bsub` reads the job commands from standard input. If the standard input is a controlling terminal, the user is prompted with "`bsub>`" for the commands of the job. The input is terminated by entering CTRL-D on a new line. You can submit multiple commands through standard input. The commands are executed in the order in which they are given. `bsub` options can also be specified in the standard input if the line begins with `#BSUB`; e.g., "`#BSUB -x`". If an option is given on both the `bsub` command line, and in the standard input, the command line option overrides the option in the standard input. The user can specify the shell to run the commands by specifying the shell path name in the first line of the standard input, such as "`#!/bin/csh`". If the shell is not given in the first line, the Bourne shell is used. The standard input facility can be used to spool a user's job script; such as "`bsub < script`". See EXAMPLES below for examples of specifying commands through standard input.

OUTPUT

If the job is successfully submitted, displays the job ID and the queue to which the job has been submitted.

EXAMPLES

% bsub sleep 100

Submit the UNIX command `sleep` together with its argument `100` as a batch job.

% bsub -q short -o my_output_file "pwd; ls"

Submit the UNIX command `pwd` and `ls` as a batch job to the queue named `short` and store the job output in `my_output` file.

% bsub -m "host1 host3 host8 host9" my_program

Submit `my_program` to run on one of the candidate hosts: host1, host3, host8 and host9.

% bsub -q "queue1 queue2 queue3" -c 5 my_program

Submit `my_program` to one of the candidate queues: `queue1`, `queue2`, and `queue3` which are selected according to the CPU time limit specified by `-c 5`.

% bsub -I ls

Submit a batch interactive job which displays the output of `ls` at the user's terminal.

% bsub -Ip vi myfile

Submit a batch interactive job to edit `myfile`.

% bsub -Is csh

Submit a batch interactive job that starts up `csh` as an interactive shell.

% bsub -b 20:00 -J my_job_name my_program

Submit `my_program` to run after 8 p.m. and assign it the job name my_job_name.

% bsub my_script

Submit `my_script` as a batch job. Since my_script is specified as a command line argument, the `my_script` file is not spooled. Later changes to the `my_script` file before the job completes may affect this job.

% bsub < default_shell_script

where default_shell_script contains:

```
sim1.exe
sim2.exe
```

The file `default_shell_script` is spooled, and the commands will be run under the Bourne shell since a shell specification is not given in the first line of the script.

```
% bsub < csh_script
```

where csh_script contains:

```
#!/bin/csh
sim1.exe
sim2.exe
```

`csh_script` is spooled and the commands will be run under `/bin/csh`.

```
% bsub -q night < my_script
```

where my_script contains:

```
#!/bin/sh
#BSUB -q test
#BSUB -o outfile -e errfile # my default stdout,
stderr files
#BSUB -m "host1 host2" # my default candidate hosts
#BSUB -f "input > tmp" -f "output << tmp"
#BSUB -D 200 -c 10/host1
#BSUB -t 13:00
#BSUB -k "dir 5"
sim1.exe
sim2.exe
```

The job is submitted to the `night` queue instead of `test`, because the command line overrides the script.

```
% bsub -b 20:00 -J my_job_name
bsub> sleep 1800
bsub> my_program
bsub> CTRL-D
```

The job commands are entered interactively.

LIMITATIONS

When using account mapping the command `bpeek(1)` will not work. File transfer via the `-f` option to `bsub(1)` requires `rcp(1)` to be working between the submission and execution hosts. Use the `-N` option to request mail, and/or the `-o` and `-e` options to specify an output file and error file, respectively.

SEE ALSO

`bjobs(1), bkill(1),bqueues(1), bhosts(1), bmgroup(1), bmod(1), bchkpnt(1), brestart(1), sh(1), getrlimit(2), sbrk(2), libckpt.a(3), lsb.users(5), lsb.queues(5), lsb.params(5), lsb.hosts(5), mbatchd(8)`

bswitch

switches unfinished jobs from one queue to another

SYNOPSIS

bswitch [**-J** *job_name*] [**-m** *host_name* | **-m** *host_group*] [**-q** *queue_name*] [**-u** *user_name* | **-u** *user_group* | **-u all**] *destination_queue* [**0**]

bswitch *destination_queue* [*job_ID* | "*job_ID*[*index_list*]"] ...

bswitch [**-h** | **-V**]

DESCRIPTION

Switches one or more of your unfinished jobs to the specified queue. LSF administrators and `root` can switch jobs submitted by other users.

By default, switches one job, the most recently submitted job, or the most recently submitted job that also satisfies other specified options (`-m`, `-q`, `-u` and `-J`). Specify `-0` to switch multiple jobs.

The switch operation can be done only if a specified job is acceptable to the new queue as if it were submitted to it, and, in case the job has been dispatched to a host, if the host can be used by the new queue. If the switch operation is unsuccessful, the job stays where it is.

If a switched job has not been dispatched, then its behavior will be as if it were submitted to the new queue in the first place.

If a switched job has been dispatched, then it will be controlled by the `loadSched` and `loadStop` vectors, PRIORITY, RUN_WINDOW and other configuration parameters of the new queue, but its nice value and resource limits will remain the same except the RUNLIMIT which is reset to the value of the new queue.

This command is useful to change a job's attributes inherited from the queue.

OPTIONS

0

(Zero). Switches multiple jobs. Switches all the jobs that satisfy other specified options (`-m`, `-q`, `-u` and `-J`).

-J *job_name*

Only switches jobs that have the specified job name.

-m *host_name* | **-m** *host_group*

Only switches jobs dispatched to the specified host or host group.

-q *queue_name*

Only switches jobs in the specified queue.

-u *user_name* | **-u** *user_group* | **-u all**

Only switches jobs submitted by the specified user or group, or all users if you specify the keyword `all`.

destination_queue

Required. Specify the queue to which the job is to be moved.

job_ID ... |"*job_ID*[*index_list*]" ...

Switches only the specified jobs.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

SEE ALSO

`bqueues(1), bhosts(1), bugroup(1), bsub(1), bjobs(1)`

LIMITATIONS

btop

moves a pending job relative to the first job in the queue

SYNOPSIS

btop *job_ID* | "*job_ID*[*index_list*]" [*position*]

btop [**-h** | **-V**]

DESCRIPTION

Changes the queue position of a pending job or a pending job array element, to affect the order in which jobs are considered for dispatch.

By default, LSF dispatches jobs in a queue in the order of their arrival (that is, first-come-first-served), subject to availability of suitable server hosts.

The `btop` command allows users and the LSF administrator to manually change the order in which jobs are considered for dispatch. Users can only operate on their own jobs, whereas the LSF administrator can operate on any user's jobs. Users can only change the relative position of their own jobs.

If invoked by the LSF administrator, `btop` moves the selected job before the first job with the same priority submitted to the queue. The positions of all users' jobs in the queue can be changed by the LSF administrator.

If invoked by a regular user, `btop` moves the selected job before the first job with the same priority submitted by the user to the queue. Pending jobs are displayed by `bjobs` in the order in which they will be considered for dispatch.

A user may use `btop` to change the dispatch order of his/her jobs scheduled using a fairshare policy. However, if a job scheduled using a fairshare policy is moved by the LSF administrator using `btop`, the job will not be subject to further fairshare scheduling unless the same job is subsequently moved by the LSF administrator using `bbot`; in this case the job will be scheduled again using the same fairshare policy (see the `FAIRSHARE` keyword in `lsb.queues`(5) and `HostPartition` keyword in `lsb.hosts` (5)).

OPTIONS

job_ID | "job_ID[index_list]"

Required. Job ID of the job or of the job array on which to operate.

For a job array, the index list, the square brackets, and the quotation marks are required. An index list is used to operate on a job array. The index list is a comma separated list whose elements have the syntax *start_index*[*-end_index*[**:***step*]] where *start_index*, *end_index* and *step* are positive integers. If the step is omitted, a step of one is assumed. The job array index starts at one. The maximum job array index is 1000. All jobs in the array share the same job_ID and parameters. Each element of the array is distinguished by its array index.

position

Optional. The *position* argument can be specified to indicate where in the queue the job is to be placed. *position* is a positive number that indicates the target position of the job from the beginning of the queue. The positions are relative to only the applicable jobs in the queue, depending on whether the invoker is a regular user or the LSF administrator. The default value of 1 means the position is before all the other jobs in the queue that have the same priority.

-h

Prints command usage to stderr and exit.

-v

Prints LSF release version to stderr and exit.

SEE ALSO

bbot(1), bjobs(1), bswitch(1)

bugroup

displays information about user groups

SYNOPSIS

bugroup [**-l**] [**-r**] [**-w**] [*user_group* ...]

bugroup [**-h** | **-V**]

DESCRIPTION

Displays user groups and user names for each group.

The default is to display information about all user groups.

OPTIONS

-l

Displays information in a long multi-line format. Also displays share distribution if shares are configured.

-r

Expands the user groups recursively. The expanded list contains only user names; it does not contain the names of subgroups. Duplicate user names are listed only once.

-w

Wide format. Displays user and user group names without truncating fields.

user_group ...

Only displays information about the specified user groups. Do not use quotes when specifying multiple user groups.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

OUTPUT

In the list of users, a name followed by a slash (/) indicates a subgroup.

FILES

User groups and user shares are defined in the configuration file `lsb.users`(5).

SEE ALSO

`lsb.users(5), bmgroup(1), busers(1)`

busers

displays information about users and user groups

SYNOPSIS

busers [*user_name* ... | *user_group* ... | **all**]

busers [**-h** | **-V**]

DESCRIPTION

Displays information about users and user groups.

By default, displays information about the user who runs the command.

OPTIONS

user_name ... | *user_group* ... | **all**

Displays information about the specified users or user groups, or about all users if you specify `all`.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

OUTPUT

A listing of the users and user groups is displayed with the following fields:

USER/GROUP

The name of the user or user group.

JL/P

The maximum number of job slots that can be processed simultaneously for the specified users on each processor. For non-preemptive scheduling, these job slots are used by running and suspended jobs or by pending jobs which have jobs slots reserved for them. For preemptive scheduling, these job slots are used by running

jobs or by pending jobs which have slots reserved for them. (see the description of PREEMPTION in `lsb.queues`(5)). This job limit is configured per processor so that multiprocessor hosts have more job slots. If the dash character (-) is displayed, there is no limit. JL/P is defined in the LSF configuration file lsb.users(5).

MAX

The maximum number of job slots that can be processed concurrently for the specified users' jobs. For non-preemptive scheduling, these job slots are used by running and suspended jobs or by pending jobs which have job slots reserved for them. For preemptive scheduling, these job slots are used by running jobs or by pending jobs which have job slots reserved for them. (see the description of PREEMPTIVE in `lsb.queues`(5)). If the character '–' is displayed, there is no limit. MAX is defined by the MAX_JOBS parameter in the configuration file `lsb.users`(5).

NJOBS

The current number of job slots used by specified users' jobs. A parallel job that is pending is counted as *n* job slots for it will use *n* job slots in the queue when it is dispatched.

PEND

The number of pending job slots used by jobs of the specified users.

RUN

The number of job slots used by running jobs of the specified users.

SSUSP

The number of job slots used by the system-suspended jobs of the specified users.

USUSP

The number of job slots used by user-suspended jobs of the specified users.

RSV

The number of job slots used by pending jobs of the specified users which have job slots reserved for them.

SEE ALSO

bugroup(1), lsb.users(5), lsb.queues(5)

ch

changes the host on which subsequent commands are to be executed

SYNOPSIS

ch [**-S**] [**-t**] [*host_name*]

ch [**-h** | **-V**]

DESCRIPTION

Changes the host on which subsequent commands are to be executed.

By default, if no arguments are specified, changes the current host to the home host, the host from which the `ch` command was issued.

By default, executes commands on the home host.

By default, shell mode support is not enabled.

By default, does not display execution time of tasks.

The `ch` command allows you to quickly change to a designated host with the same execution environment. A simple shell is started that delivers all subsequent commands (except built-in commands) to the designated host for execution.

When the simple shell starts, it is in the current working directory and has the same command execution environment as that of the parent shell. Every remotely dispatched command is executed with the same environment as that on the home host. The syntax of the `ch` command is similar to that of the Bourne shell. However, there are some important differences.

The ampersand `&` following a command line (representing a background job in the Bourne shell) is ignored by `ch`. You can submit background jobs in `ch` with the built-in `post` command and bring them into the foreground with the built-in `contact` command (see below for details).

`ch` recognizes a `~` (tilde) as a special path name. If a `~` (tilde) is followed by a space, tab, new line or `/` (slash) character, then the `~` character is translated into the user's home directory. Otherwise, the `~` is translated as the home directory of the user name given by the string following the `~` character. Pipelines, lists of commands and redirection of standard input/output are all handled by invoking `/bin/sh`.

The following sequence of commands illustrates the behavior of the `ch` command. For example, the user is currently on `hostA`:

```
% ch hostB
hostB> ch hostC
hostC> ch
hostA> ... ...
```

OPTIONS

`-s`

Starts remote tasks with shell mode support. Shell mode support is required for running interactive shells or applications which redefine the `CTRL-C` and `CTRL-Z` keys (for example, `jove`).

`-t`

Turns on the timing option. The amount of time each subsequent command takes to execute is displayed.

`host_name`

Executes subsequent commands on the specified host.

`-h`

Prints command usage to stderr and exits.

`-V`

Prints LSF release version to stderr and exits.

USAGE

The `ch` command interprets the following built-in commands:

`cd` [*`directory_name`*]

Changes the current working directory to the specified directory. If a directory is not specified, changes to the user's home directory by default.

`ch` [*`host_name`*]

Changes the current working host to the specified host. If a host is not specified, changes to the home host by default.

post [*command* [*argument* ...]]

Posts the specified command for execution in the background on the current working host. `ch` assigns a unique task ID to this command and displays this ID, then continues to interact with the user. However, the output of background jobs may disturb the screen. You can post multiple commands on one host or on different hosts. When a previously posted command is completed, `ch` reports its status to the standard error. If a command is not specified, `ch` displays all currently running background commands.

contact *task_ID*

Brings a previously posted background command into the foreground. *task_ID* is the ID returned by the `post` command. Standard input is now passed to this foreground command. You cannot put an active foreground job into the background. A command that has been brought into the foreground with the `contact` command cannot be put back into the background.

exit

Exits `ch` if there are no posted commands running. Typing an EOF character (usually `CTRL-D` but may be set otherwise, see `stty(1)`) forces `ch` to exit; uncompleted posted commands are killed.

SEE ALSO

`lsrun(1), rsh(1), stty(1)`

LIMITATIONS

Currently, the `ch` command does not support script, history, nor alias.

The `ch` prompt is always the *current working host:current working directory* followed by a `>` (right angle bracket) character. If the `ch` session is invoked by a shell that supports job control (such as `tcsh` or `ksh`), `CTRL-Z` suspends the whole `ch` session. The exit status of a command line is printed to stderr if the status is non-zero.

lsacct

displays accounting statistics on finished RES tasks in the LSF system

SYNOPSIS

lsacct [**-l**] [**-C** *time0*,*time1*] [**-S** *time0*,*time1*] [**-f** *logfile_name*] [**-m** *host_name*] [**-u** *user_name* ... | **-u all**] [*pid* ...]

lsacct [**-h** | **-V**]

DESCRIPTION

Displays statistics on finished tasks run through RES. When a remote task completes, RES logs task statistics in the task log file.

By default, displays accounting statistics for only tasks owned by the user who invoked the `lsacct` command.

By default, displays accounting statistics for tasks executed on all hosts in the LSF system.

By default, displays statistics for tasks logged in the task log file currently used by RES: `LSF_RES_ACCTDIR/lsf.acct.`*host_name* `or /tmp/lsf.acct.`*host_name* (see `lsf.acct`(5)).

If `-l` is not specified, the default is to display the fields in SUMMARY only (see OUTPUT).

The RES on each host writes its own accounting log file. These files can be merged using the `lsacctmrg` command to generate statistics for the entire LSF cluster.

All times are reported in seconds. All sizes are reported in kilobytes.

OPTIONS

`-l`

Per-task statistics. Displays statistics about each task. See OUTPUT for a description of information that is displayed.

`-C` *time0,time1*

Displays accounting statistics for only tasks that completed or exited during the specified time interval.

The time format is the same as in `bhist`(1).

-S *time0,time1*

Displays accounting statistics for only tasks that began executing during the specified time interval.

The time format is the same as in **bhist**(1).

-f *logfile_name*

Searches the specified task log file for accounting statistics. Specify either an absolute or a relative path.

Useful for analyzing old task log files or files merged with the **lsacctmrg** command.

-m *host_name ...*

Displays accounting statistics for only tasks executed on the specified hosts.

If a list of hosts is specified, host names must be separated by spaces and enclosed in quotation marks (") or (').

-u *user_name ...* | **-u all**

Displays accounting statistics for only tasks owned by the specified users, or by all users if the keyword **all** is specified.

If a list of users is specified, user names must be separated by spaces and enclosed in quotation marks (") or ('). You can specify both user names and user IDs in the list of users.

pid ...

Displays accounting statistics for only tasks with the specified *pid*. This option overrides all other options except for **-l**, **-f**, **-h**, **-V**.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

OUTPUT

SUMMARY (default format)

Overall statistics for tasks.

The total, average, maximum and minimum resource usage statistics apply to all specified tasks.

The following fields are displayed:

Total number of tasks

Total number of tasks including tasks completed successfully and total number of exited tasks.

Time range of started tasks

Start time of the first and last task selected.

Time range of ended tasks

Completion or exit time of the first and last task selected.

Resource usage of tasks selected

See `getrusage` (2).

CPU time

Total CPU time consumed by the task.

Page faults

Number of page faults.

Swaps

Number of times the process was swapped out.

Blocks in

Number of input blocks.

Blocks out

Number of output blocks.

Messages sent

Number of System VIPC messages sent.

Messages rcvd

Number of IPC messages received.

Voluntary cont sw

Number of voluntary context switches.

Involuntary con sw

Number of involuntary context switches.

Turnaround

Elapsed time from task execution to task completion.

Per Task Statistics (-l)

In addition to the fields displayed by default in SUMMARY, displays the following fields for each task:

Starting time

Time the task started.

User and host name

User who submitted the task, host from which the task was submitted, in the format *user_name@host*.

PID

UNIX process ID of the task.

Execution host

Host on which the command was run.

Command line

Complete command line that was executed.

CWD

Current working directory of the task.

Completion time

Time at which the task completed.

Exit status

UNIX exit status of the task.

FILES

Reads `lsf.acct.`*host_name*

SEE ALSO

`lsf.acct(5), lsacctmrg(1), res(8), bhist(1)`

lsacctmrg

merges task log files

SYNOPSIS

lsacctmrg [**-f**] *logfile_name ... target_logfile_name*

lsacctmrg [**-h** | **-V**]

DESCRIPTION

Merges specified task log files into the specified target file in chronological order according to completion time.

All files must be in the format specified in `lsf.acct` (see `lsf.acct`(5)).

OPTIONS

`-f`

Overwrites the target file without prompting for confirmation.

`logfile_name ...`

Specify log files to be merged into the target file, separated by spaces. Specify either an absolute or a relative path.

`target_logfile_name`

Specify the file into which all log files are to be merged. Specify either an absolute or a relative path. The target file cannot be part of the files to be merged.

`-h`

Prints command usage to stderr and exits.

`-v`

Prints LSF release version to stderr and exits.

SEE ALSO

`lsf.acct(5), res(8)`

lsadmin

administrative tool for LSF

SYNOPSIS

lsadmin *subcommand*
lsadmin [**-h** | **-V**]

SUBCOMMAND LIST

ckconfig [**-v**]
reconfig [**-f**] [**-v**]
limstartup [**-f**] [*host_name* ... | **all**]
limshutdown [**-f**] [*host_name* ... | **all**]
limrestart [**-v**] [**-f**] [*host_name* ... | **all**]
limlock [**-l** *time_seconds*]
limunlock
limdebug [**-c** *class_name* ...] [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**] [*host_name*]
limtime [**-l** *timing_level*] [**-f** *logfile_name*] [**-o**] [*host_name*]
resstartup [**-f**] [*host_name* ... | **all**]
resshutdown [**-f**] [*host_name* ... | **all**]
resrestart [**-f**] [*host_name* ... | **all**]
reslogon [*host_name* ... | **all**] [**-c** *cpu_time*]
reslogoff [*host_name* ... | **all**]
resdebug [**-c** *class_name* ...] [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**] [*host_name*]
restime [**-l** *timing_level*] [**-f** *logfile_name*] [**-o**] [*host_name*]
help [*subcommand* ...]
quit

DESCRIPTION

This command can only be used by LSF administrators.

`lsadmin` is a tool that executes privileged commands to control LIM and RES operations in an LSF cluster.

If no subcommands are supplied for `lsadmin`, `lsadmin` prompts for subcommands from the standard input.

For subcommands for which multiple host names or host groups can be specified, do not enclose the multiple names in quotation marks.

OPTIONS

subcommand

Executes the specified subcommand. See Usage section.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version and exits.

USAGE

ckconfig [-v]

Checks LSF configuration files.

-v

Displays detailed messages about configuration file checking.

reconfig [-f] [-v]

Restarts LIMs on all hosts in the cluster. You should use `reconfig` after changing configuration files. The configuration files are checked before all LIMs in the cluster are restarted. If the configuration files are not correct, reconfiguration will not be initiated.

-f

Disables user interaction and forces LIM to restart on all hosts in the cluster if no fatal errors are found. This option is useful in batch mode.

-v

Displays detailed messages about configuration file checking.

limstartup [-f] [*host_name* ... |all]

Starts up the LIM on the local host if no arguments are specified.

Starts up LIMs on the specified hosts or on all hosts in the cluster if the word `all` is the only argument provided. You will be asked for confirmation.

Only `root` and users listed in `lsf.sudoers`(5) in the parameter LSF_STARTUP_USERS can start up LIM as `root`. These users must also be able to use `rsh` on all LSF hosts without having to type in passwords. If permission to start up LIMs as `root` is not configured, `limstartup` will start up LIMs as yourself after your confirmation.

-f

Disables interaction and does not ask for confirmation for starting up LIMs.

limshutdown [-f] [*host_name* ... | all]

Shuts down LIM on the local host if no arguments are supplied.

Shuts down LIMs on the specified hosts or on all hosts in the cluster if the word `all` is specified. You will be asked for confirmation.

-f

Disables interaction and does not ask for confirmation for shutting down LIMs.

limrestart [-v] [-f] [*host_name* ... | all]

Restarts LIM on the local host if no arguments are supplied.

Restarts LIMs on the specified hosts or on all hosts in the cluster if the word `all` is specified. You will be asked for confirmation.

`limrestart` should be used with care. Do not make any modifications until all the LIMs have completed the startup process. If you execute `limrestart` *host_name*`...` to restart some of the LIMs after changing the configuration files, but other LIMs are still running the old configuration, confusion will arise among these LIMs. To avoid this situation, use `reconfig` instead of `limrestart`.

-v

Displays detailed messages about configuration file checking.

-f

Disables user interaction and forces LIM to restart if no fatal errors are found. This option is useful in batch mode. `limrestart -f all` is the same as `reconfig -f`.

limlock [-l *time_seconds*]

Locks LIM on the local host until it is explicitly unlocked if no time is specified.When a host is locked, LIM's load status becomes **lockU**. No job will be sent to a locked host by LSF.

-l *time_seconds*

The host is locked for the specified time in seconds. This is useful if a machine is running an exclusive job requiring all the available CPU time and/or memory.

limunlock

Unlocks LIM on the local host.

resstartup [-f] [*host_name ...* **| all]**

Starts up RES on the local host if no arguments are specified.

Starts up RESs on the specified hosts or on all hosts in the cluster if the word **all** is specified. You will be asked for confirmation.

For the **all** and **-f** options, only **root** and users defined by the LSF_STARTUP_USERS parameter in **lsf.sudoers**(5) can use this option to start up RES as **root**, and those users must be able to use **rsh** on all LSF hosts. For **root** installation to work properly, **lsadmin** must be installed as a **setuid** to **root** program.

-f

Disables interaction and does not ask for confirmation for starting up RESs.

resshutdown [-f] [*host_name ...* **| all]**

Shuts down RES on the local host if no arguments are specified.

Shuts down RESs on the specified hosts or on all hosts in the cluster if the word **all** is specified. You will be asked for confirmation.

If RES is running, it will keep running until all remote tasks exit.

-f

Disables interaction and does not ask for confirmation for shutting down RESs.

resrestart [**-f**] [*host_name* ... | **all**]

Restarts RES on the local host if no arguments are specified.

Restarts RESs on the specified hosts or on all hosts in the cluster if the word **all** is specified. You will be asked for confirmation.

If RES is running, it will keep running until all remote tasks exit. While waiting for remote tasks to exit, another RES is restarted to serve the new queries.

-f

Disables interaction and does not ask for confirmation for restarting RESs.

reslogon [*host_name* ... | **all**] [**-c** *cpu_time*]

Logs all tasks executed by RES on the local host if no arguments are specified.

Logs tasks executed by RESs on the specified hosts or on all hosts in the cluster if **all** is specified.

RES will write the task's resource usage information into the log file **lsf.acct.***host_name*. The location of the log file is determined by LSF_RES_ACCTDIR defined in **lsf.conf**. If LSF_RES_ACCTDIR is not defined, or RES cannot access it, the log file will be created in **/tmp** instead.

-c *cpu_time*

Logs only tasks that use more than the specified amount of CPU time. The amount of CPU time is specified by *cpu_time* in milliseconds.

reslogoff [*host_name* ... | **all**]

Turns off RES task logging on the local host if no arguments are specified.

Turns off RES task logging on the specified hosts or on all hosts in the cluster if **all** is specified.

limdebug [**-c** "*class_name ...*"]
[**-l** *debug_level*] [**-f** *logfile_name*]
[**-o**] ["*host_name ...*"]

Sets the message log level for LIM to include additional information in log files. You must be `root` or the LSF administrator to use this command.

If the command is used without any options, the following default values are used:

class_name = 0 (no additional classes are logged)

debug_level = 0 (LOG_DEBUG level in parameter LSF_LOG_MASK)

logfile_name = current LSF system log file in the directory specified by LSF_LOGDIR in the format *daemon_name*.`log`.*host_name*

host_name= local host (host from which command was submitted)

-c "*class_name ...*"

Specify software classes for which debug messages are to be logged. If a list of classes is specified, they must be enclosed in quotation marks and separated by spaces.

Possible classes:

LC_AFS - Log AFS messages

LC_AUTH - Log authentication messages

LC_CHKPNT - log checkpointing messages

LC_COMM - Log communication messages

LC_DCE - Log messages pertaining to DCE support

LC_EXEC - Log significant steps for job execution

LC_FILE - Log file transfer messages

LC_HANG - Mark where a program might hang

LC_LICENCE - Log licence management messages

LC_MULTI - Log messages pertaining to MultiCluster

LC_PIM - Log PIM messages

LC_SCHED - Log JobScheduler messages

LC_SIGNAL - Log messages pertaining to signals

LC_TRACE - Log significant program walk steps

LC_XDR - Log everything transferred by XDR

Default: 0 (no additional classes are logged)

Note: Classes are also listed in `lsf.h`.

-l *debug_level*

Specify level of detail in debug messages. The higher the number, the more detail that is logged. Higher levels include all lower levels.

Possible values:

0 - LOG_DEBUG level in parameter LSF_LOG_MASK in `lsf.conf`.

1 - LOG_DEBUG1 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 LOG_DEBUG1, and LOG_DEBUG levels.

2 - LOG_DEBUG2 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 LOG_DEBUG1, and LOG_DEBUG levels.

3 - LOG_DEBUG3 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2, LOG_DEBUG1, and LOG_DEBUG levels.

Default: 0 (LOG_DEBUG level in parameter LSF_LOG_MASK)

-f *logfile_name*

Specify the name of the file into which debugging messages are to be logged. A file name with or without a full path may be specified.

If a file name without a path is specified, the file will be saved in the directory indicated by the parameter LSF_LOGDIR in `lsf.conf`.

The name of the file that will be created will have the following format:

logfile_name`.`*daemon_name*`.log.`*host_name*

If the specified path is invalid, on UNIX, the log file is created in the `/tmp` directory. On Windows NT, no log file is created.

If LSF_LOGDIR is not defined, daemons log to the `syslog` facility.

Default: current LSF system log file in the directory specified by LSF_LOGDIR in the format *daemon_name*`.log.`*host_name*`.`

`-o`

Turns off temporary debug settings and reset them to the daemon starting state. The message log level is reset back to the value of LSF_LOG_MASK and classes are reset to the value of LSF_DEBUG_RES, LSF_DEBUG_LIM.

Log file is reset back to the default log file.

`"host_name ..."`

Sets debug settings on the specified host or hosts.

Default: local host (host from which command was submitted)

`resdebug` `[-c "`*`class_name`*`"] [-l `*`debug_level`*`] [-f `*`logfile_name`*`] [-o] ["`*`host_name ...`*`"]`

Sets the message log level for RES to include additional information in log files. You must be the LSF administrator to use this command, not `root`.

See description of `limdebug` for an explanation of options.

`limtime` `[-l `*`timing_level`*`] [-f `*`logfile_name`*`] [-o] ["`*`host_name ...`*`"]`

Sets timing level for LIM to include additional timing information in log files. You must be `root` or the LSF administrator to use this command.

If the command is used without any options, the following default values are used:

timing_level = no timing information is recorded

logfile_name = current LSF system log file in the directory specified by LSF_LOGDIR in the format *daemon_name*`.log.`*host_name*

host_name = local host (host from which command was submitted)

-l *timing_level*

Specifies detail of timing information that is included in log files. Timing messages indicate the execution time of functions in the software and are logged in milliseconds.

Valid values: 1 | 2 | 3 | 4 | 5

The higher the number, the more functions in the software that are timed and whose execution time is logged. The lower numbers include more common software functions. Higher levels include all lower levels.

Default: undefined (no timing information is logged)

-f *logfile_name*

Specify the name of the file into which timing messages are to be logged. A file name with or without a full path may be specified.

If a file name without a path is specified, the file will be saved in the directory indicated by the parameter LSF_LOGDIR in `lsf.conf`.

The name of the file that will be created will have the following format:

logfile_name`.`*daemon_name*`.log.`*host_name*

If the specified path is invalid, on UNIX, the log file is created in the `/tmp` directory. On Windows NT, no log file is created.

If LSF_LOGDIR is not defined, daemons log to the syslog facility.

Note: Both timing and debug messages are logged in the same files.

Default: current LSF system log file in the directory specified by LSF_LOGDIR in the format *daemon_name*`.log.`*host_name*.

-o

Turns off temporary timing settings and resets them to the daemon starting state. The timing level is reset back to the value of the parameter for the corresponding daemon (LSF_TIME_LIM, LSF_TIME_RES).

Log file is reset back to the default log file.

"host_name ..."

Sets the timing level on the specified host or hosts.

Default: local host (host from which command was submitted)

restime [**-l** *timing_level*] [**-f** *logfile_name*] [**-o**] [*"host_name ..."*]

Sets timing level for RES to include additional timing information in log files. You must be the LSF administrator can use this command, not `root`.

See description of **limtime** for an explanation of options.

help [*subcommand ...*] | **?** [*subcommand ...*]

Displays the syntax and functionality of the specified commands. The commands must be explicit to **lsadmin**.

From the command prompt, you may use **help** or **?**.

quit

Exits the **lsadmin** session.

SEE ALSO

`ls_limcontrol(3), ls_rescontrol(3), ls_readconfenv(3), ls_gethostinfo(3), ls_connect(3), ls_initrex(3), lsf.conf(5), lsf.sudoers(5), lsf.acct(5), bmgroup(1), busers(1)lsreconfig(8), lslockhost(8), lsunlockhost(8)`

lsclusters

displays configuration information about LSF clusters

SYNOPSIS

lsclusters [**-l**] [*cluster_name ...*]

lsclusters [**-h** | **-V**]

DESCRIPTION

Displays configuration information about LSF clusters.

By default, returns information about the local cluster and all other clusters of which the local cluster is aware (all clusters defined in the `RemoteClusters` section of `lsf.cluster.`*cluster_name* if that section exists, otherwise all clusters defined in `lsf.shared`).

OPTIONS

`-l`

Long format. Displays additional information.

`cluster_name ...`

Only displays information about the specified clusters.

`-h`

Prints command usage to stderr and exits.

`-V`

Prints LSF release version to stderr and exits.

OUTPUT

Default Output

The information includes: cluster name, cluster master host, primary cluster administrator's login name, total number of hosts in the cluster, and the number of server hosts in the cluster.

A listing of the clusters is displayed with the following fields:

CLUSTER_NAME

The name of the cluster.

STATUS

The current status of the cluster. Possible values are:

ok

The cluster is in normal load sharing state, and will exchange load information with the local cluster.

unavail

The cluster is unavailable.

MASTER_HOST

The name of the cluster's master host.

ADMIN

The user account name of the cluster's primary LSF administrator.

HOSTS

Number of LSF hosts in the cluster.

SERVERS

Number of LSF server hosts in the cluster.

Long Format (-l)

If this option is specified, the command will also list available resource names, host types, host models and cluster administrator's login names, and whether local cluster accepts or sends interactive jobs to this cluster.

SEE ALSO

`lsfintro(1), ls_info(3), ls_policy(3), ls_clusterinfo(3)`
`lsf.cluster(5)`

lseligible

displays whether a task is eligible for remote execution

SYNOPSIS

lseligible [**-r**] [**-q**] [**-s**] *task*

lseligible [**-h** | **-V**]

DESCRIPTION

Displays whether the specified task is eligible for remote execution.

By default, only tasks in the remote task list are considered eligible for remote execution.

OPTIONS

-r

Remote mode. Considers eligible for remote execution any task not included in the local task list.

-q

Quiet mode. Displays only the resource requirement string defined for the task. The string ELIGIBLE or NON-ELIGIBLE is omitted.

-s

Silent mode. No output is produced. The `-q` and `-s` options are useful for shell scripts which operate by testing the exit status (see DIAGNOSTICS).

task

Specify a command.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

OUTPUT

If the task is eligible, the string ELIGIBLE followed by the resource requirements associated with the task are printed to stdout. Otherwise, the string NON-ELIGIBLE is printed to stdout.

If `lseligible` prints ELIGIBLE with no resource requirements, the task has the default requirements of CPU consumption and memory usage.

SEE ALSO

`ls_eligible(3), lsrtasks(1), lsf.task(5)`

DIAGNOSTICS

`lseligible` has the following exit statuses:

0 Task is eligible for remote execution

1 Command is to be executed locally

-1 Syntax errors

-10 A failure is detected in the LSF system

lsfmon

installs or uninstalls LSF Monitor

SYNOPSIS

lsfmon -install

lsfmon -remove

DESCRIPTION

Installs or uninstalls LSF Monitor in an existing cluster.

LSF Monitor runs on Windows NT and allows you to use Windows NT Performance Monitor to chart information about the LSF cluster.

The LSF Monitor service runs under the account of an LSF cluster administrator.

OPTIONS

`-install`

Installs LSF Monitor on the host.

`-remove`

Removes LSF Monitor from the host.

lsfrestart

restarts LIM, RES, SBD and MBD on all hosts in the cluster

SYNOPSIS

lsfrestart [**-f** | **-h** | **-V**]

DESCRIPTION

This command can only be used by root or users listed in lsf.sudoers.

Restarts LIM, RES, SBD and MBD, in that order, on all hosts in the local cluster.

By default, prompts for confirmation of the next operation if an error is encountered.

In order to be able to control all daemons in the cluster:

- The file `/etc/lsf.sudoers` has to be set up properly. See `lsf.sudoers(5)`.

- You have to be able to run the `rsh` command across all LSF hosts without having to enter a password. Refer to your operating system documentation for information about configuring the `rsh` command.

OPTIONS

`-f`

Force mode. Continues to restart daemons even if an error is encountered.

`-h`

Prints command usage to stderr and exits.

`-v`

Prints LSF release version to stderr and exits.

SEE ALSO

`lsadmin(8), badmin(8), lsfshutdown(8)`

lsfsetup

runs `lsfsetup`, the LSF installation and configuration utility

SYNOPSIS

LSF_SERVERDIR/lsfsetup

DESCRIPTION

This command can only be used by LSF administrators or root.

You must be root to run `lsfsetup` for installation operations. You must be an LSF administrator to run `lsfsetup` for modifying configuration options.

`lsfsetup` is a command to run `lsfsetup`, a utility to install and upgrade LSF software and licenses, and to maintain LSF configuration files.

The `lsfsetup` utility is distributed with the LSF distribution files and is located in the setup directory created when you uncompress and extract installation script distribution files.

At installation, `lsfsetup` creates the `lsf.conf` file (see `lsf.conf(5)`) and relies on this file for all subsequent operations.

After installation, `lsfsetup` is located in LSF_SERVERDIR and symbolically linked to LSF_BINDIR (see `lsf.conf(5)`).

`lsfsetup` is menu-based and provides extensive help information. If you have any problems understanding a particular selection, type `?` at the prompt for help.

SEE ALSO

`lsf.conf(5)`, *LSF UNIX Installation Guide*

lsfshutdown

shuts down LIM, RES, SBD and MBD on all hosts in the cluster

SYNOPSIS

lsfshutdown [**-f** | **-h** | **-V**]

DESCRIPTION

This command can only be used by root or users listed in lsf.sudoers.

Shuts down SBD, RES, LIM, and MBD, in that order, on all hosts.

By default, prompts for confirmation of the next operation if an error is encountered.

In order to be able to control all daemons in the cluster:

- The file `/etc/lsf.sudoers` has to be set up properly. See `lsf.sudoers(5)`.

- You have to be able to run the `rsh` command across all LSF hosts without having to enter a password. Refer to your operating system documentation for information about configuring the `rsh` command.

OPTIONS

-f

Force mode. Continues to shut down daemons even if an error is encountered.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

SEE ALSO

`lsadmin(8), badmin(8), lsfrestart(8)`

lsfstartup

starts LIM, RES, SBD, and MBD on all hosts in the cluster

SYNOPSIS

lsfstartup [**-f**]

lsfstartup [**-h** | **-V**]

DESCRIPTION

This command can only be used by root or users listed in lsf.sudoers.

Starts up LIM, RES, SBD, and MBD, in that order, on all hosts.

By default, prompts for confirmation of the next operation if an error is encountered.

If LSF daemons are already running, use `lsfrestart` instead, or use `lsfshutdown` and shut down the running daemons before you use `lsfstartup`.

In order to be able to control all daemons in the cluster:

- The file `/etc/lsf.sudoers` has to be set up properly. See `lsf.sudoers(5)`.

- You have to be able to run the `rsh` command across all LSF hosts without having to enter a password. Refer to your operating system documentation for information about configuring the `rsh` command.

OPTIONS

`-f`

Force mode. Continues to start daemons even if an error is encountered.

`-h`

Prints command usage to stderr and exits.

`-v`

Prints LSF release version to stderr and exits.

SEE ALSO

lsadmin(8), badmin(8), lsf.sudoers(5), lsfshutdown(8), lsfrestart(8)

lsgrun

executes a task on a set of hosts

SYNOPSIS

lsgrun [**-i**] [**-p** | **-P** | **-S**] [**-v**]
-f *host_file* | **-m** *host_name ...* | **-n** *num_procesors* [**-R** "*res_req*"]
[*command* **[***argument ...***]**]

lsgrun [**-h** | **-V**]

DESCRIPTION

Executes a task on the specified hosts.

By default, `lsgrun` is not interactive.

By default, the specified task will be executed sequentially on hosts with full pseudo `tty` support.

By default, `lsgrun` does not create a pseudo-terminal.

By default, LSF uses as many processors as available to run the specified task.

By default, the resource requirement for host selection is `r15s:pg.`

By default, the prompt `Command>` is displayed to allow users to type in a command (task) terminated by a `CTRL-D` or `EOF`. The command is then executed on the specified hosts.

`lsgrun` is useful for fast global operations such as starting daemons, replicating files to or from local disks, looking for processes running on all hosts, checking who is logged in on each host, and so on. The hosts can be specified using a host file, a list of host names or by letting the system select the hosts.

OPTIONS

`-i`

Interactive operation mode. You are asked whether the task will be executed on all hosts. If you answer `y`, the task is started on all specified hosts; otherwise, you are asked to specify hosts interactively.

-p

Parallel run mode. Executes the task on all hosts simultaneously and without pseudo `tty` support.

If this option is specified and the `-P` option is specified, the `-P` option is ignored.

This option is useful for fast start-up of tasks. However, any output from remote tasks will arrive at the terminal in arbitrary order, depending on task execution speeds on individual hosts.

-P

Creates a pseudo-terminal. This is necessary to run programs requiring a pseudo-terminal (for example, `vi`).

-S

Creates a pseudo-terminal with shell mode support.

Shell mode support is required for running interactive shells or applications which redefine the `CTRL-C` and `CTRL-Z keys` (such as `jove`).

-v

Verbose mode. Displays the name of the host or hosts running the task.

-f *host_file*

Either `-f` *host_file*, `-m` *host_name* or `-n` *num_processors* is required.

Executes the task on all hosts listed in the *host_file*.

Specify a file that contains a list of host names. Host names must be separated by white space characters (for example, SPACE, TAB, and NEWLINE).

This option is exclusive of options `-n`, `-R`, and `-m`.

-m *host_name ...*

Either `-f` *host_file*, `-m` *host_name* or `-n` *num_processors* is required.

Executes the task on all specified hosts.

Specify hosts on which to execute the task. If multiple host names are specified, the host names must be enclosed by " or ' and separated by white space.

This option is exclusive of options `-n`, `-R`, and `-f`.

-n *num_processors*

Either **-f** *host_file*, **-m** *host_name* or **-n** *num_processors* is required.

Executes the task on hosts with the required number of available processors.

One host may be used to start several tasks if the host is multiprocessor. This option can be used together with option **-R** to select desired hosts.

This option is exclusive of options **-m** and **-f**.

-R "*res_req*"

Executes the task on hosts with the required resource requirements.

Specify the resource requirement expression for host selection. The resource requirement will be used to choose from all hosts with the same host type as the local host, unless a "**type == value**" exists in *res_req* to specify otherwise.

This option can be used together with option **-n** to choose a specified number of processors to run the task.

command [*argument ...*]

Specify the command to execute. This must be the last argument on the command line.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

SEE ALSO

lsfintro(1), lsrun(1), lsplace(1)

DIAGNOSTICS

Exit status is 0 if all commands are executed correctly.

Otherwise, the exit status is the first non-zero status returned by a remotely executed task. **lsgrun** will execute the task on all hosts even if some have non-zero exit status.

Exit status is -10 if a problem is detected in LSF.

lshosts

displays hosts and their static resource information

SYNOPSIS

lshosts [**-w** | **-l**] [**-R** "*res_req*"] [*host_name*] ...

lshosts -s [*shared_resource_name* ...]

lshosts [**-h** | **-V**]

DESCRIPTION

Displays static resource information about hosts.

By default, returns the following information: host name, host type, host model, CPU factor, number of CPUs, total memory, total swap space, whether or not the host is a server host, and static resources. Displays information about all hosts in thecluster . See `lsf.cluster`(5).

The `-s` option displays information about the static shared resources and their associated hosts.

OPTIONS

`-w`

Displays host information in wide format. Fields are displayed without truncation.

`-l`

Displays host information in a long multi-line format. In addition to the default fields, displays information about the maximum `/tmp` space, the number of local disks, the execution priority for remote jobs, load thresholds, and run windows.

`-R "`*`res_req`*`"`

Only displays information about the hosts that satisfy the resource requirement expression. For more information about resource requirements, see `lsfintro`(1). The size of the resource requirement string is limited to 512 bytes.

LSF supports ordering of resource requirements on all load indices, including external load indices, either static or dynamic.

host_name.....

Only displays information about the specified hosts. Do not use quotes when specifying multiple hosts.

-s [*shared_resource_name ...*]

Displays information about the specified resources. The resources must be static shared resources. Returns the following information: the resource names, the values of the resources, and the resource locations. If no shared resource is specified, then displays information about all shared resources.

-h

Prints command usage to stderr and exits.

-V

Prints the LSF release version to stderr and exits.

OUTPUT

Host-Based Default

Displays the following fields:

HOST_NAME

The name of the host. The host name is truncated if too long.

type

The host type. The host type is truncated if too long.

model

The host model. The host model is truncated if too long.

cpuf

The CPU factor. The CPU factor is used to scale the CPU load value so that differences in CPU speeds are considered. The faster the CPU, the larger the CPU factor.

The CPU factor of a host with an `unknown` host type is 1.0.

ncpus

The number of CPUs.

maxmem

The total memory.

maxswp

The total swap space.

server

"Yes" if the host is a server host.

RESOURCES

The available Boolean resources denoted by resource names, and the values of external numeric and string static resources. See `lsf.cluster`(5), and `lsf.shared`(5) on how to configure external static resources.

Host Based -l Option

In addition to the above fields, the `-l` option also displays the following:

ndisks

The number of local disks.

maxtmp

The maximum `/tmp` space in megabytes configured on a host.

rexpri

The remote execution priority.

RUN_WINDOWS

The time periods during which the host is open for sharing loads from other hosts. (See `lsf.cluster`(5).)

LOAD_THRESHOLDS

The thresholds for scheduling interactive jobs. If a load threshold is exceeded, the host status is changed to "busy." See `lsload`(1).

Resource-Based -s Option

Displays the static shared resources. Each line gives the value and the associated hosts for the static shared resource. See `lsf.shared`(5), and `lsf.cluster`(5) on how to configure static shared resources.

The following fields are displayed:

RESOURCE

The name of the resource.

VALUE

The value of the static shared resource.

LOCATION

The hosts that are associated with the static shared resource.

SEE ALSO

`lsfintro(1), ls_info(3), ls_policy(3), ls_gethostinfo(3), lsf.cluster(5), lsf.shared(5)`

lsid

displays the current LSF version number, the cluster name, and the master host name

SYNOPSIS

lsid [**-h** | **-V**]

DESCRIPTION

Displays the current LSF version number, the cluster name, and the master host name.

The master host is dynamically selected from all hosts in the cluster.

OPTIONS

-h

Prints command usage to stderr and exits.

-v

Prints LSF release version to stderr and exits.

FILES

The host names and cluster name are defined in `lsf.cluster.`*cluster_name* and `lsf.shared`, respectively.

SEE ALSO

`ls_getclustername(3), ls_getmastername(3), lsinfo(1)`

lsinfo

displays load sharing configuration information

SYNOPSIS

lsinfo [**-l**] [**-m** | **-M**] [**-r**] [**-t**] [*resource_name* ...]

lsinfo [**-h** | **-V**]

DESCRIPTION

By default, displays all load sharing configuration information including resource names and their meanings, host types and models, and associated CPU factors known to the system.

By default, displays information about all resources. Resource information includes resource name, resource type, description, and the default sort order for the resource.

You can use resource names in task placement requests.

Use this command with options to selectively view configured resources, host types, and host models.

OPTIONS

`-l`

Displays resource information in a long multi-line format. Additional parameters are displayed including whether a resource is built-in or configured, and whether the resource value changes dynamically or is static. If the resource value changes dynamically then the interval indicates how often it is evaluated.

`-m`

Displays only information about host models that exist in the cluster.

`-M`

Displays information about all host models in the file `lsf.shared`.

`-r`

Displays only information about configured resources.

-t

Displays only information about configured host types. See `lsload(1)` and `lshosts(1)`.

resource_name ...

Displays only information about the specified resources.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

SEE ALSO

`lsfintro(1), lshosts(1), lsload(1), lsf.shared(5), ls_info(3), ls_policy(3)`

lsload

displays load information for hosts

SYNOPSIS

lsload [**-l**] [**-N** | **-E**] [**-I** *load_index*[**:***load_index*] ...] [**-n** *num_hosts*] [**-R** *res_req*] *host_name* ... ...

lsload -s [*resource_name* ...]

lsload [**-h** | **-V**]

DESCRIPTION

Displays load information for hosts. Load information can be displayed on a per-host basis, or on a per-resource basis.

By default, displays load information for all hosts in the local cluster, per host.

By default, displays raw load indices.

By default, load information for resources is displayed according to CPU and paging load.

OPTIONS

-l

Long format. Displays load information without truncation along with additional fields for I/O and external load indices.

This option overrides the index names specified with the `-I` option.

-N

Displays normalized CPU run queue length load indices (see `lsfintro(1)`).

-E

Displays effective CPU run queue length load indices (see `lsfintro(1)`). Options `-N` and `-E` are mutually exclusive.

-I *load_index*[:*load_index*] ...

Displays only load information for the specified load indices. Load index names must be separated by a colon (for example, `r1m:pg:ut`).

-n *num_hosts*

Displays only load information for the requested number of hosts. Information for up to *num_hosts* hosts that best satisfy the resource requirements is displayed.

-R *res_req*

Displays only load information for hosts that satisfy the specified resource requirements. See `lsinfo(1)` for a list of built-in resource names.

Load information for the hosts is sorted according to load on the specified resources.

If *res_req* contains special resource names, only load information for hosts that provide these resources is displayed (see `lshosts(1)` to find out what resources are available on each host).

If one or more host names are specified, only load information about the hosts that satisfy the resource requirements is displayed.

host_name ...

Displays only load information for the specified hosts.

-s [*resource_name* ...]

Displays information for all dynamic shared resources configured in the cluster.

If resources are specified, only displays information for the specified resources. *resource_name* must be a dynamic shared resource name.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

OUTPUT

HOST-BASED OUTPUT (Default output)

Numeric dynamic non-shared resources are displayed. The selection and order sections of *res_req* control for which hosts information is displayed and how they are ordered (see `lsfintro(1)`).

The displayed default load information includes the following fields:

HOST_NAME

Standard host name used by LSF, typically an Internet domain name with two components.

status

Status of the host. A minus sign (-) may precede the status, indicating that the Remote Execution Server (RES) on the host is not running.

Possible statuses are:

ok

The host is in normal load sharing state and can accept remote jobs.

busy

The host is overloaded because some load indices exceed configured thresholds. Load index values that caused the host to be busy are preceded by an asterisk (`*`). Built-in load indices include `r15s`, `r1m`, `r15m`, `ut`, `pg`, `io`, `ls`, `it`, `swp`, `mem` and `tmp` (see below). External load indices are configured in the file `lsf.cluster.`*cluster_name* (see `lsf.cluster(5)`).

r15s

The 15-second exponentially averaged CPU run queue length.

r1m

The 1-minute exponentially averaged CPU run queue length.

r15m

The 15-minute exponentially averaged CPU run queue length.

ut

The CPU utilization exponentially averaged over the last minute, between 0 and 1.

pg

The memory paging rate exponentially averaged over the last minute, in pages per second.

io

The disk I/O rate exponentially averaged over the last minute, in KB per second (this is only available when the -l option is specified).

ls

The number of current login users.

it

On UNIX, the idle time of the host (keyboard not touched on all logged in sessions), in minutes.

On Windows NT, the `it` index is based on the time a screen saver has been active on a particular host.

swp

The amount of swap space available, in megabytes.

mem

The amount of available memory, in megabytes.

tmp

The amount of free space in `/tmp`, in megabytes.

external_index

Any site-configured global external load indices (see `lim(8)`). Available only when the `-l` option or the `-I` option with the index name is used, and only if defined in the `lsf.cluster.`*cluster_name* (see `lsf.cluster(5)`) configuration file. Note that *external_index* should not contain shared resources.

lockW

The host is locked by its run window. Run windows for a host are specified in the configuration file (see `lsf.conf(5)`) and can be displayed by `lshosts`. A locked host will not accept load shared jobs from other hosts.

lockU

The host is locked by the LSF administrator or `root`.

unavail

The host is down or the Load Information Manager (LIM) on the host is not running.

unlicensed

The host does not have a valid LSF license.

RESOURCE-BASED OUTPUT (lsload -s)

Displays information about dynamic shared resources. Each line gives the value and the associated hosts for an instance of the resource. See `lim(8)`, and `lsf.cluster(5)` for information on configuring dynamic shared resources.

The displayed information consists of the following fields:

RESOURCE

Name of the resource.

VALUE

Value for an instance of the resource.

LOCATION

Hosts associated with the instance of the resource.

EXAMPLES

```
% lsload -R "select[r1m<=0.5 && swp>=20 && type==ALPHA]"
```

OR, in restricted format:

```
% lsload -R r1m=0.5:swp=20:type=ALPHA
```

Displays the load of ALPHA hosts with at least 20 megabytes of swap space, and a 1-minute run queue length less than 0.5.

```
% lsload -R "select[(1-swp/maxswp)<0.75] order[pg]"
```

Displays the load of the hosts whose swap space utilization is less than 75%. The resulting hosts are ordered by paging rate.

```
% lsload -I r1m:ut:io:pg
```

Displays the 1-minute CPU raw run queue length, the CPU utilization, the disk I/O and paging rates for all hosts in the cluster.

```
% lsload -E
```

Displays the load of all hosts, ordered by `r15s:pg`, with the CPU run queue lengths being the effective run queue lengths (see `lsfintro(1)`).

```
% lsload -s verilog_license
```

Displays the value and location of all the `verilog_license` dynamic shared resource instances.

SEE ALSO

`lsfintro(1), lim(8), lsf.cluster(5), lsplace(1), lshosts(1), lsinfo(1), lslockhost(8), ls_load(3)`

DIAGNOSTICS

Exit status is -10 if an LSF problem is detected or a bad resource name is specified.

Exit status is -1 if a bad parameter is specified, otherwise `lsload` returns 0.

lsloadadj

adjusts load indices on hosts

SYNOPSIS

lsloadadj [**-R** *res_req*] [*host_name*[**:***num_task*] ...]

lsloadadj [**-h** | **-V**]

DESCRIPTION

Adjusts load indices on hosts. This is useful if a task placement decision is made outside LIM by another application.

By default, assumes tasks are CPU-intensive and memory-intensive. This means the CPU and memory load indices are adjusted to a higher number than other load indices.

By default, adjusts load indices on the local host, the host from which the command was submitted.

By default, starts 1 task.

Upon receiving a load adjustment request, LIM temporarily increases the load on hosts according to resource requirements. This helps LIM avoid sending too many jobs to the same host in quick succession. The inflated load decays over time before the real load produced by the dispatched task is reflected in LIM's load information.

`lsloadadj` adjusts all indices with the exception of `ls` (login sessions), `it` (idle time), `r15m` (15-minute run queue length) and external load indices.

OPTIONS

`-R` *`res_req`*

Specify resource requirements for tasks. Only the resource usage section of the resource requirement string is considered (see `lsfintro(1)`). This is used by LIM to determine by how much individual load indices are to be adjusted.

For example, if a task is swap-space-intensive, load adjustment on the `swp` load index is higher; other indices are increased only slightly.

***host_name*[:*num_task*] ...**

Specify a list of hosts for which load is to be adjusted. *num_task* indicates the number of tasks to be started on the host.

-h

Prints command usage to stderr and exits.

-v

Prints LSF release version to stderr and exits.

EXAMPLES

```
% lsloadadj -R "rusage[swp=20:mem=10]"
```

Adjusts the load indices `swp` and `mem` on the host from which the command was submitted.

SEE ALSO

`lsinfo(1), lsplace(1), lsload(1), ls_loadadj(3)`

DIAGNOSTICS

Returns -1 if a bad parameter is specified; otherwise returns 0.

lslockhost

alias for `lsadmin limlock`

DESCRIPTION

This command can only be used by LSF administrators.

Provided for backward compatibility.

lslogin

remotely logs in to a lightly loaded host

SYNOPSIS

lslogin [**-v**] [**-m** "*host_name ...*" ""] [**-R** "*res_req*"] [*rlogin_options*]

lslogin [**-h** | **-V**]

DESCRIPTION

Remotely logs in to a lightly loaded host.

By default, `lslogin` selects the least loaded host, with few users logged in, and remotely logs in to that host using the UNIX `rlogin` command.

OPTIONS

-v

Displays the name of the host to which `lslogin` remotely logs you in.

-m "*host_name ...*" ""

Remotely logs in to the specified host.

-R "*res_req*"

Remotely logs in to a host that meets the specified resource requirement. The resource requirement expression restricts the set of candidate hosts and determines the host selection policy.

For a complete explanation of resource requirement expressions, see `lsfintro(1)`. To find out what resources are configured in your system, use `lsinfo(1)` and `lshosts(1)`.

rlogin_options

Specify remote login options passed to the `rlogin` command.

If remote execution fails, `lslogin` will log in locally only if the local host also satisfies required resources; otherwise, log in will fail.

-h

Prints command usage to stderr and exits.

-v

Prints LSF release version to stderr and exits.

EXAMPLE

```
% lslogin -R "select[it>1 && bsd]"
```

Remotely logs in to a host that has been idle for at least 1 minute, that runs BSD UNIX, and is lightly loaded both in CPU resources and the number of users logged in.

SEE ALSO

`lsfintro(1), ls_placereq(3), rlogin(1)`

DIAGNOSTICS

Because `lslogin` passes all unrecognized arguments to `rlogin`, incorrect options usually cause the `rlogin` usage message to be displayed rather than the `lslogin` usage message.

lsltasks

displays or updates a user's local task list

SYNOPSIS

lsltasks [+ *task_name* ... | – *task_name* ...]

lsltasks [-**h** | -**V**]

DESCRIPTION

Displays or updates a user's local task list in `$HOME/.lsftask`.

When no options are specified, displays tasks listed in the system task file `lsf.task` and the user's task file `.lsftask`.

If there is a conflict between the system task file `lsf.task` and the user's task file `.lsftask`, the user's task file overrides the system task file.

Tasks in the local task list are not eligible for remote execution, either because they are trivial tasks or because they need resources on the local host.

OPTIONS

+ *task_name*

If + is specified and the specified task names are not already in the file `.lsftask` in the user's home directory, adds the task names to the file with a plus sign (+) preceding them.

If any of the task names are already in the `.lsftask` file, the actual action depends on the entry in the file. If the entry starts with a + or nothing, replaces the entry with the specified content; if the entry starts with a minus sign (-), deletes the entry from the `.lsftask` file.

- *task_name*

If – is specified and specified task names are not already in the file `.lsftask` in the user's home directory, adds the task names to the file with a – preceding the task name.

If any of the task names are already in the `.lsftask` file, the actual action depends on the entry in the file. If the entry starts with a –, no operation is done; if the entry starts with a +, deletes the entry from the `.lsftask` file.

`-h`

Prints command usage to stderr and exits.

`-v`

Prints LSF release version to stderr and exits.

EXAMPLES

```
% lsltasks + foo
```

Adds the command `foo` to the local task list.

FILES

Reads the system task file `lsf.task`, and the user task file `.lsftask` in the user's home directory. See `lsf.task(5)` for more details.

The system and user task files contain two sections, one for the remote task list, the other for the local task list. The local tasks section starts with `Begin LocalTasks` and ends with `End LocalTasks`. Each line in the section is an entry consisting of a task name.

A plus sign (+) or a minus sign (–) can optionally precede each entry. If no + or – is specified, then + is assumed.

SEE ALSO

`lsfintro(1), lseligible(1), ls_task(3), lsrtasks(1), lsf.task(5), ls_eligible(3)`

lsmake

runs make tasks in parallel

SYNOPSIS

lsmake [**-c** *num_tasks*] [**-F** *res_req*] [**-m** "*host_name* ..."] [**-E**] [**-G**] [**-M**] [**-V**] [*makeoption* ...] [*target* ...]

lsmake [**-c** *num_tasks*] [**-F** *res_req*] [**-j** *max_processors*] [**-P** *minutes*] [**-R** *res_req*] [**-E**] [**-G**] [**-M**] [**-V**] [*makeoption* ...] [*target* ...]

DESCRIPTION

Runs make tasks in parallel on LSF hosts. Sets the environment variables on the remote hosts when `lsmake` first starts up.

By default, uses the local host, uses only one processor, starts only one task in each processor, and processes submakes sequentially.

`lsmake` is a modified version of `GNU make`. All the options provided by `GNU make` are valid with `lsmake`.

OPTIONS

`-c` *`num_tasks`*

Starts the specified number of tasks concurrently on each processor. If you specify too many tasks, you could overload a host.

`-F` *`res_req`*

Temporarily reduces the number of tasks running when the load on the network file server exceeds the specified resource requirements. This might also reduce the number of processors used. The number of tasks is increased again when the load on the network file server is below the specified resource requirements.

The network file server is considered to be the host mounting the current working directory on the local host. If this machine is not in the local cluster, `-F` is ignored.

`-m` **`"`***`host_name ...`***`"`**

Uses the specified hosts. Specify a host name multiple times to use multiple processors on that host.

-j *max_processors*

Uses multiple processors. Specify the maximum number of processors to use. Uses all of the available processors if fewer processors are available.

When you specify `-j` and `-R` together, automatically selects processors on the best available hosts that satisfy the resource requirements. The job fails if no suitable host is found.

When you specify `-j` but not `-R`, automatically selects processors on the best available hosts that are the same host type as the local host. The local host itself can be selected.

-P *minutes*

Periodically reselects the best available processors. After the processor has been used for the specified number of minutes, it might be replaced if a better processor is available.

This is is useful for long-running makes.

-R *res_req*

Uses only hosts that satisfy the specified resource requirements.

When you specify `-R` but not `-j`, uses one processor on one host that satisfies the resource requirements.

-E

Sets the environment variables for every task sent remotely.

This is necessary when make files change or override the environment variables they inherit at startup.

-G

Enables debugging.

-M

Processes submakes in parallel. Some makefiles may not work correctly when run in parallel through LSF Make.

To use this feature, build each submake as a separate target in your makefile. Specify the make command for each submake with the built-in $(MAKE) macro. Makefiles that depend on sequential processing might have to be modified further.

For more information, see the LSF Make documentation.

-v

Verbose mode. Prints the names of the hosts used.

makeoption ...

Specifies `GNU Make` options. See `gmake(1)` for details.

target ...

Specifies targets to make.

SEE ALSO

`lsfintro(1), lstcsh(1), gmake(1)`

For a complete description of how to use LSF Make, see the LSF Make documentation.

LIMITATIONS

If a submake in a makefile specifies options which are specific to `lsmake` they are ignored. Only the command line options are used.

When determining where to start tasks, `lsmake` consults the local task list (see `lsf.task`(5)). If the task is found in the local task list, then it will be started on the local host. The resource requirements of tasks in the remote task list are not considered when dispatching tasks.

lsmon

displays load information for LSF hosts and periodically updates the display

SYNOPSIS

lsmon [**-N** | **-E**] [**-n** *num_hosts*] [**-R** *res_req*] [**-I** *index_list*] [**-i** *interval*] [**-L** *file_name*] [*host_name* ...]

lsmon [**-h** | **-V**]

DESCRIPTION

`lsmon` is a full-screen LSF monitoring utility that displays and updates load information for hosts in a cluster.

By default, displays load information for all hosts in the cluster, up to the number of lines that will fit on-screen.

By default, displays raw load indices.

By default, load information is sorted according to CPU and paging load.

By default, load information is updated every 10 seconds.

OPTIONS

`-N`

Displays normalized CPU run queue length load indices (see `lsfintro(1)`).

`-E`

Displays effective CPU run queue length load indices (see `lsfintro(1)`). Options `-N` and `-E` are mutually exclusive.

`-n` *`num_hosts`*

Displays only load information for the requested number of hosts. Information for up to *num_hosts* hosts that best satisfy resource requirements is displayed.

-R *res_req*

Displays only load information for hosts that satisfy the specified resource requirements. See `lsinfo`(1) for a list of built-in resource names.

Load information for the hosts is sorted according to load on the specified resources.

If *res_req* contains special resource names, only load information for hosts that provide these resources is displayed (see `lshosts(1)` to find out what resources are available on each host).

If one or more host names are specified, only load information for the hosts that satisfy the resource requirements is displayed.

-I *index_list*

Displays only load information for the specified load indices. Load index names must be separated by a colon (for example, `r1m:pg:ut`).

If the index list *index_list* is too long to fit in the screen of the user who invoked the command, the output is truncated. For example, if the invoker's screen is 80 characters wide, then up to 10 load indices are displayed.

-i *interval*

Sets how often load information is updated on-screen, in seconds.

-L *file_name*

Saves load information in the specified file while it is displayed on-screen.

If you do not want load information to be displayed on your screen at the same time, use **lsmon -L** *file_name* **< /dev/null**. The format of the file is described in `lim.acct`(5).

host_name ...

Displays only load information for the specified hosts.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

USAGE

You can use the following commands while `lsmon` is running:

[**^L** | **i** | **n** | **N** | **E** | **R** | **q**]

^L

Refreshes the screen.

i

Prompts you to input a new update interval.

n

Prompts you to input a new number of hosts to display.

N

Toggles between displaying raw CPU run queue length load indices and normalized CPU run queue length load indices.

E

Toggles between displaying raw CPU run queue length load indices and effective CPU run queue length load indices.

R

Prompts you to input new resource requirements.

q

Quits `lsmon`.

OUTPUT

The following fields are displayed by default.

HOST_NAME

Name of specified hosts for which load information is displayed, or if resource requirements were specified, name of hosts that satisfied the specified resource requirement and for which load information is displayed.

status

Status of the host. A minus sign (-) may precede the status, indicating that the Remote Execution Server (RES) on the host is not running.

Possible statuses are:

ok

The host is in normal load sharing state and can accept remote jobs.

busy

The host is overloaded because some load indices exceed configured thresholds. Load index values that caused the host to be busy are preceded by an asterisk (`*`). Built-in load indices include `r15s`, `r1m`, `r15m`, `ut`, `pg`, `io`, `ls`, `it`, `swp`, `mem` and `tmp` (see below). External load indices are configured in the file `lsf.cluster.`*cluster_name* (see `lsf.cluster(5)`).

lockW

The host is locked by its run window. Run windows for a host are specified in the configuration file (see `lsf.conf(5)`) and can be displayed by `lshosts`. A locked host will not accept load shared jobs from other hosts.

lockU

The host is locked by the LSF administrator or `root`.

unavail

The host is down or the Load Information Manager (LIM) on the host is not running.

unlicensed

The host does not have a valid LSF license.

r15s

The 15-second exponentially averaged CPU run queue length.

r1m

The 1-minute exponentially averaged CPU run queue length.

r15m

The 15-minute exponentially averaged CPU run queue length.

ut

The CPU utilization exponentially averaged over the last minute, between 0 and 1.

pg

The memory paging rate exponentially averaged over the last minute, in pages per second.

ls

The number of current login users.

it

On UNIX, the idle time of the host (keyboard not touched on all logged in sessions), in minutes.

On Windows NT, the `it` index is based on the time a screen saver has been active on a particular host.

tmp

The amount of free space in `/tmp`, in megabytes.

swp

The amount of currently available swap space, in megabytes.

mem

The amount of currently available memory, in megabytes.

SEE ALSO

`lsfintro(1), lshosts(1), lsinfo(1), lsload(1), lslockhost(8), lim.acct(5), ls_load(3)`

DIAGNOSTICS

Specifying an invalid resource requirement string while `lsmon` is running (via the `R` option) causes `lsmon` to exit with an appropriate error message.

`lsmon` exits if it does not receive a reply from LIM within the update interval.

lspasswd

registers user passwords in LSF on Windows NT

SYNOPSIS

lspasswd [**-u** *user_name*]

DESCRIPTION

Registers user passwords in LSF on Windows NT.

By default, if no options are specified, the password applies to the user who issued the command.

Only the LSF administrator can enter passwords for other users.

Users must update the password maintained by LSF if they change their Windows NT account password.

Passwords are Windows NT account passwords and are saved in the LSF database. LSF uses the passwords to start jobs on behalf of the user.

`lspasswd` communicates with LSF services on the local machine to store the password. The password is stored in encrypted format and the password database is protected by Windows NT file access permissions.

OPTIONS

-u *user_name*

Specify the user whose password you want to change. Only the LSF administrator can enter passwords for other users.

LIMITATIONS

You must issue the `lspasswd` command from an LSF server host. You cannot issue this command from an LSF client host.

lsplace

displays hosts available to execute tasks

SYNOPSIS

lsplace [**-L**] [**-n** *minimum* | **-n 0**] [**-R** *res_req*] [**-w** *maximum* | **-w 0**] [*host_name ...*]

lsplace [**-h** | **-V**]

DESCRIPTION

Displays hosts available for the execution of tasks, and temporarily increases the load on these hosts (to avoid sending too many jobs to the same host in quick succession). The inflated load will decay slowly over time before the real load produced by the dispatched task is reflected in the LIM's load information. Host names may be duplicated for multiprocessor hosts, to indicate that multiple tasks can be placed on a single host.

By default, displays only one host name.

By default, uses LSF default resource requirements.

OPTIONS

-L

Attempts to place tasks on as few hosts as possible. This is useful for distributed parallel applications in order to minimize communication costs between tasks.

-n *minimum* | **-n 0**

Displays at least the specified number of hosts. Specify 0 to display as many hosts as possible.

Prints `Not enough host(s) currently eligible` and exits with status 1 if the required number of hosts holding the required resources cannot be found.

-R *res_req*

Displays only hosts with the specified resource requirements.

-w *maximum* | **-w 0**

Displays no more than the specified number of hosts. Specify 0 to display as many hosts as possible.

host_name ...

Displays only hosts that are among the specified hosts.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

EXAMPLES

lsplace is mostly used in backquotes to pick out a host name which is then passed to other commands. The following example issues a command to display a lightly loaded HPPA-RISC host for your program to run on:

```
% lsrun -m 'lsplace -R hppa' myprogram
```

The **-w** and **-n** options can be combined to specify the upper and lower bounds in processors to be returned, respectively. For example, the command **lsplace -n 3 -w 5** returns at least 3 and not more than 5 host names.

SEE ALSO

lsinfo(1), ls_placereq(3), lsload(1), lsrun(1)

DIAGNOSTICS

lsplace returns 1 if insufficient hosts are available. The exit status is -10 if a problem is detected in LSF, -1 for other errors, otherwise 0.

lsrcp

remotely copies files using LSF

SYNOPSIS

lsrcp [**-a**] *source_file target_file*

lsrcp [**-h** | **-V**]

DESCRIPTION

Remotely copies files using LSF.

`lsrcp` is an LSF-enabled remote copy program that transfers a single file between hosts in an LSF cluster. `lsrcp` uses RES on an LSF host to transfer files. If LSF is not installed on a host or if RES is not running then `lsrcp` uses `rcp` to copy the file.

To use `lsrcp`, you must have read access to the file being copied.

Both the source and target file must be owned by the user who issues the command.

`lsrcp` uses `rcp` to copy a source file to a target file owned by another user. See `rcp(1)` and LIMITATIONS below for details.

OPTIONS

-a

Appends *source_file* to *target_file*.

source_file target_file

Specify an existing file on a local or remote host that you want to copy, and a file to which you want to copy the source file.

File format is as follows:

[[*user_name*@][*host_name*]:][*path*/]*file_name*

user_name

Login name to be used for accessing files on the remote host. If *user_name* is not specified, the name of the user who issued the command is used.

host_name

Name of the remote host on which the file resides. If *host_name* is not specified, the local host, the host from which the command was issued is used.

path

Absolute path name or a path name relative to the login directory of the user. Shell file name expansion is not supported on either the local or remote hosts. Only single files can be copied from one host to another.

Use "\" to transfer files from a Windows host to another Windows host. For example:

```
c:\share>lsrcp file1 hostA:c:\temp\file2
```

Use "/" to transfer files from a UNIX host to a UNIX host. For example:

```
% lsrcp file1 hostD:/home/usr2/test/file2
```

Always use "/" to transfer files from a UNIX host to a Windows host, or from a Windows host to a UNIX host. This is because the operating system interprets "\" and `lsrcp` will open the wrong files.

For example, to transfer a file from UNIX to a Windows host:

```
% lsrcp file1 hostA:/c:/temp/file2
```

For example, to transfer a file from Windows to a UNIX host:

```
c:\share>lsrcp file1 hostD:/home/usr2/test/file2
```

file_name

Name of source file. File name expansion is not supported.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

EXAMPLES

```
% lsrcp myfile @hostC:/home/usr/dir1/otherfile
```

Copies file `myfile` from the local host to file `otherfile` on `hostC`.

% lsrcp user1@hostA:/home/myfile user1@hostB:otherfile

Copies the file `myfile` from `hostA` to file `otherfile` on `hostB`.

% lsrcp -a user1@hostD:/home/myfile /dir1/otherfile

Appends the file `myfile` on `hostD` to the file `otherfile` on the local host.

% lsrcp /tmp/myfile user1@hostF:~/otherfile

Copies the file `myfile` from the local host to file `otherfile` on `hostF` in user1's home directory.

SEE ALSO

`rsh(1), rcp(1), lsfintro(1), res(8)`

DIAGNOSTICS

`lsrcp` attempts to copy *source_file* to *target_file* using RES. If RES is down or fails to copy the *source_file*, `lsrcp` will use either `rsh` when the `-a` option is specified, or `rcp` when `-a` is not specified.

LIMITATIONS

File transfer using `lscrp` is not supported in the following contexts:

- If LSF account mapping is used; `lsrcp` fails when running under a different user account

- On LSF client hosts. LSF client hosts do not run RES, so `lsrcp` cannot contact RES on the submission host

- Third party copies. `lsrcp` does not support third party copies, when neither source nor target file are on the local host. In such a case `rcp` or `rsh` will be used. If the *target_file* exists, `lsrcp` preserves the modes; otherwise, `lsrcp` uses the *source_file* modes modified with the `umask` (see `umask(2)`) of the source host.

You can do the following:

`rcp` on UNIX- if `lsrcp` cannot contact RES on the submission host, it attempts to use `rcp` to copy the file. You must set up the `/etc/hosts.equiv` or `HOME/.rhosts` file in order to use `rcp`. See the `rcp(1)` and `rsh(1)` manual pages for more information on using the `rcp` command.

You can replace `lsrcp` with your own file transfer mechanism as long as it supports the same syntax as `lsrcp`. This might be done to take advantage of a faster interconnection network, or to overcome limitations with the existing `lsrcp`. SBD looks for the `lsrcp` executable in the LSF_BINDIR directory as specified in the `lsf.conf` file.

lsreconfig

alias for `lsadmin reconfig`

DESCRIPTION

This command can only be used by LSF Administrators.

Provided for backward compatibility.

lsrtasks

displays or updates a user's remote task list

SYNOPSIS

lsrtasks [**+** *task_name*[*/res_req*] ... | **–** *task_name*[*/res_req*] ...]

lsrtasks [**-h** | **-V**]

DESCRIPTION

Displays or updates a user's remote task list in `$HOME/.lsftask`.

When no options are specified, displays tasks listed in the system task file `lsf.task` and the user's task file `.lsftask`.

If there is a conflict between the system task file `lsf.task` and the user's task file `.lsftask`, the user's task file overrides the system task file.

Tasks in the remote task list are eligible for remote execution. You can associate resource requirements with each task name. Eligibility of tasks not specified in a task list for remote execution depends on the operation mode: local or remote. In local mode, tasks are not eligible for remote execution; in remote mode, tasks are eligible. You can specify the operation mode when deciding the eligibility of a task (see `lseligible(1)`, and `ls_eligible(3)`).

OPTIONS

+ *task_name*`[`*/res_req*`] ...`

If plus sign (+) is specified and the specified task names are not already in the file `.lsftask` in the user's home directory, adds the task names to the file with a + sign preceding them.

If any of the task names are already in the `.lsftask` file, the actual action depends on the entry in the file. If the entry starts with a + or nothing, replaces the entry with the specified content; if the entry starts with a minus sign (–), deletes the entry from the `.lsftask` file.

Remote tasks can have a resource requirement expression associated with them, separated by a backslash (/). See `ls_task(3)`.

- *task_name*[/*res_req*] ...

If – is specified and specified task names are not already in the file `.lsftask` in the user's home directory, adds the task names to the file with a – preceding the task name.

If any of the task names are already in the `.lsftask` file, the actual action depends on the entry in the file. If the entry starts with a –, no operation is done; if the entry starts with a +, deletes the entry from the `.lsftask` file.

Remote tasks can have a resource requirement expression associated with them, separated by a backslash /. See `ls_task`(3).

-h

Prints command usage to stderr and exits.

-v

Prints LSF release version to stderr and exits.

EXAMPLES

```
% lsrtasks + task1 task2/"select[cpu && mem]" - task3
```

or in restricted form:

```
% lsrtasks + task1 task2/cpu:mem - task3
```

Adds the command `task1` to the remote task list with no resource requirements, adds `task2` with the resource requirement `cpu:mem`, and removes `task3` from the remote task list.

FILES

Reads the system task file `lsf.task`, and the user task file `.lsftask` in the user's home directory. See `lsf.task`(5) for more details.

The system and user task files contain two sections, one for the remote task list, the other for the local task list. The remote tasks section starts with `Begin RemoteTasks` and ends with `End RemoteTasks`. Each line in the section is an entry consisting of a task name.

A plus sign + or a minus sign – can optionally precede each entry. If no + or – is specified, then + is assumed.

SEE ALSO

lsfintro(1), lseligible(1), ls_task(3), lsltasks(1), lsf.task(5), ls_eligible(3)

lsrun

runs an interactive task through LSF

SYNOPSIS

lsrun [**-l**] [**-P**] [**-S**] [**-v**] [**-m** "*host_name* ..."] [**-R** "*res_req*"] *command* [*argument* ...]

lsrun [**-h** | **-V**]

DESCRIPTION

Submits a task to LSF for execution.

By default, `lsrun` first tries to obtain resource requirement information from the remote task list to find an eligible host. (See `lseligible`(1) and `ls_task`(3).) Otherwise, `lsrun` runs the task on a host that is of the same host type (or architecture) as the submission host. If several hosts of the same architecture are available, the host with the lowest CPU and memory load is selected.

By default, if execution fails and the local host satisfies resource requirements, LSF runs the task locally.

By default, `lsrun` does not create a pseudo-terminal when running the task.

OPTIONS

`-l`

If execution on another host fails, runs the task locally.

`-P`

Creates a pseudo-terminal when starting the task. This is necessary in order to run programs that require a pseudo-terminal (for example, `vi`).

`-S`

Creates a pseudo-terminal with shell mode support when starting the task. Shell mode support is required for running interactive shells or applications which redefine the `CTRL-C` and `CTRL-Z` keys (for example, `jove`).

-v

Displays the name of the host running the task.

-m "*host_name ...*"

The execution host must be one of the specified hosts.

If a single host is specified, all resource requirements are ignored.

If multiple hosts are specified and you do not use the **-R** option, the execution host must satisfy the resource requirements in the remote task list (see **lsrtasks**(1)). If none of the specified hosts satisfy the resource requirements, the task will not run.

-R "*res_req*"

Runs the task on a host that meets the specified resource requirement. The size of the resource requirement string is limited to 512 bytes. For a complete explanation of resource requirement expressions, see **lsfintro**(1). To find out what resources are configured in your system, use **lsinfo**(1) and **lshosts**(1).

LSF supports ordering of resource requirements on all load indices, including external load indices, either static or dynamic.

If the **-m** option is specified with a single host name, the **-R** option is ignored.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

USAGE

You can use **lsrun** together with other utility commands such as **lsplace**(1), **lsload**(1), **lsloadadj**(1), and **lseligible**(1) to write load sharing applications in the form of UNIX shell scripts.

lsrun supports interactive job control. Suspending **lsrun** suspends both the task and **lsrun**, and continuing **lsrun** continues the task.

The **-n** option of **rsh**(1) can be simulated by redirecting input from **/dev/null**. For example:

```
lsrun cat </dev/null &
```

SEE ALSO

`rsh(1), lsfintro(1), ls_rexecv(3), lsplace(1), lseligible(1), lsload(1), lshosts(1), lsrtasks(1), lsf.cluster(5)`

DIAGNOSTICS

`lsrun` exits with status -10 and prints an error message to stderr if a problem is detected in LSF and the task is not run.

The exit status is -1 and an error message is printed to stderr if a system call fails or incorrect arguments are specified.

Otherwise, the exit status is the exit status of the task.

lstcsh

load sharing `tcsh` for LSF

SYNOPSIS

lstcsh [*tcsh_options*] [**-L**] [*argument* ...]

DESCRIPTION

`lstcsh` is an enhanced version of `tcsh`. `lstcsh` behaves exactly like `tcsh`, except that it includes a load sharing capability with transparent remote job execution for LSF.

By default, a `lstcsh` script is executed as a normal `tcsh` script with load sharing disabled.

If a command line is considered eligible for remote execution, LSF selects a suitable host— typically a powerful and/or lightly loaded host that can execute the command line correctly—and sends the command line to that host.

You can restrict who can use `@` for host redirection in `lstcsh` with the parameter LSF_SHELL_AT_USERS in `lsf.conf`.

Remote Hosts

`lstcsh` provides a high degree of network transparency. Command lines executed on remote hosts behave the same as they do on the local host. The remote execution environment is designed to mirror the local one as closely as possible by using the same values for environment variables, terminal setup, current working directory, file creation mask, and so on. Each modification to the local set of environment variables is automatically reflected on remote hosts.

Shell variables, nice values, and resource limits are not automatically propagated to remote hosts.

Job Control

Job control in `lstcsh` is exactly the same as in `tcsh` except for remote background jobs. `lstcsh` numbers background jobs separately for each of the hosts that are used to execute them. The output of the built-in command `job` lists background jobs together with their execution hosts.

To bring a remote background job to the foreground, the host name must be specified together with an at sign (@), as in the following example:

```
fg %2 @hostA
```

Similarly, the host name must be specified when killing a remote job. For example:

```
kill %2 @hostA
```

OPTIONS

tcsh_options

`lstcsh` accepts all the options used by `tcsh`. See `tcsh(1)` for the meaning of specific options.

`-L`

Executes a script with load sharing enabled.

There are three ways to run a `lstcsh` script with load sharing enabled:

- Execute the script with the `-L` option

- Use the built-in command `source` to execute the script

- Insert "`#!/local/bin/lstcsh -L`" as the first line of the script (assuming you install `lstcsh` in `/local/bin`).

Using `@` or `lsmode` (see below) in a script will not enable load sharing if the script has not been executed using one of these three ways.

USAGE

In addition to the built-in commands in `tcsh`, `lstcsh` provides the following built-in commands:

lsmode [**on** | **off**] [**local** | **remote**] [**@**] [**v** | **-v**] [**e** | **-e**] [**t** | **-t**] [**connect** [*host_name* ...]] [**lsrtasks** [*lsrtasks_options*]] [**lsltasks** [*lsltasks_options*]] [**jobs**]

`on | off`

Turns load sharing on or off. When off, you can send a command line to a remote host only if forced eligibility is specified with `@`.

`local | remote`

Sets operation mode of `lstcsh`.

The default is `local`.

`local`

Local operation mode. This is the default mode.

In this mode, a command line is eligible for remote execution only if all the specified tasks are present in the remote task list in the user's tasks file `$HOME/.lsftask`, or if `@` is specified on the command line to force specified tasks to be eligible for remote execution.

Tasks in the local task list must be executed locally.

The local mode of operation is conservative, and can fail to take advantage of the performance benefits and load balancing advantages of LSF.

The way `lstcsh` handles tasks that are not present in the remote task list nor in the local task list, depends on the mode of operation of `lstcsh` (local or remote).

`remote`

Remote operation mode.

In this mode, a command line is considered eligible for remote execution only if none of the specified tasks are present in the local task list in the user's tasks file `$HOME/.lsftask.`

Tasks in the remote list can be executed remotely.

The remote mode of operation is aggressive, and promotes extensive use of LSF.

The way `lstcsh` handles tasks that are not present in the remote task list nor in the local task list, depends on the mode of operation of `lstcsh` (local or remote).

`@`

Specify `@` to explicitly specify the eligibility of a command for remote execution.

The `@` may be anywhere in the command line except in the first position (which is used to set the value of shell variables).

There are several ways to use `@`:

@

Specify @ followed by nothing to indicate the command line is eligible for remote execution.

@ *host_name*

Specify @ followed by a host name to force the command line to be executed on that host.

Host names and the reserved word `local` following @ can all be abbreviated as long as they do not cause ambiguity.

@ local

Specify @ followed by the reserved word `local` to force the command line to executed on the local host.

@ */res_req*

Specify @ followed by / and a resource requirement string to indicate the command is eligible for remote execution, and that the specified resource requirements must be used instead of those in the remote task list.

When specifying resource requirements following the @ it is necessary to use / only if the first requirement characters specified are also the first characters of a host name.

e | -e

Turns eligibility verbose mode on (`e`) or off (`-e`).

If eligibility verbose mode is on, `lstcsh` shows whether the command is eligible for remote execution, and displays the resource requirement used if the command is eligible.

The default is off.

v | -v

Turns task placement verbose mode on (`v`) or off (`-v`). If verbose mode is on, `lstcsh` displays the name of the host on which the command is run if the command is not run on the local host.

The default is on.

t | -t

Turns wall clock timing on (`t`) or off (`-t`).

If timing is on, the actual response time of the command is displayed. This is the total elapsed time in seconds from the time you submit the command to the time the prompt comes back.

This time includes all remote execution overhead. The `csh` time built-in does not include the remote execution overhead.

This is an impartial way of comparing the response time of jobs submitted locally or remotely, because all the load sharing overhead is included in the displayed elapsed time.

The default is off.

`connect` [*`host_name ...`*]

Establishes connections with specified remote hosts. If no hosts are specified, lists all the remote hosts to which an `lstcsh` connection has been established.

A plus sign (+) with a remote host indicates that a server-shell has also been started on it.

`lsrtasks` [**`+`** *`task_name`*[*`/res_req ...`*] | **`-`** *`task_name`*[*`/res_req ...`*]]

Displays or update a user's remote task list in the user's task list `$HOME/.lsftask`.

This command has the same function as the external command `lsrtasks`, except that the modified remote task list takes effect immediately for the current `lstcsh` session.

See `lsrtasks(1)` for more details.

`lsltasks` [**`+`** *`task_name ...`* | **`-`** *`task_name ...`*]

Displays or update a user's local task list in the user's task list `$HOME/.lsftask`.

This command has the same function as the external command `lsltasks`, except that the modified local task list takes effect immediately for the current `lstcsh` session.

See `lsltasks`(1) for more details.

`jobs`

Lists background jobs together with the execution hosts. This break of transparency is intentional in order to provide you with more control over your background jobs.

FILES

There are three optional configuration files for `lstcsh`:

`.shrc`

`.hostrc`

`.lsftask`

The `.shrc` and `.hostrc` files are used by `lstcsh` alone, whereas `.lsftask` is used by LSF to determine general task eligibility.

~/.shrc

Use this file when you want an execution environment on remote hosts that is different from that on the local host. This file is sourced automatically on a remote host when a connection is established. For example, if the remote host is of different type, you may need to run a version of the executable for that particular host type, therefore it may be necessary to set a different path on the remote host.

~/.hostrc

Use this file to indicate a list of host names to which the user wants to be connected (asynchronously in the background) at `lstcsh` startup time. This saves the time spent in establishing the connections dynamically during execution of shell commands. Once a connection is set up, you can execute further remote commands on those connected hosts with very little overhead.

~/.lsftask

Use this file to specify lists of remote and local tasks that you want to be added to the respective system default lists. Each line of this file is of the form *task_name/res_req*, where *task_name* is the name of a task, and *res_req* is a string specifying the resource requirements of the task. If *res_req* is not specified, the command is executed on machines of the same type as the local host.

SEE ALSO

`csh(1), tcsh(1), lsrtasks(1), lsltasks(1), lseligible(1), lsinfo(1), lsload(1)`

LIMITATIONS

Type-ahead for the `next` command is discarded when a job is executing in the foreground on a remote host.

It is not possible to provide input data to load sharing shell scripts (that is, shell scripts whose content is load shared).

The `lstcsh` is fully compatible with tcsh 6.03 7-bit mode. Any feature that is not included in `tcsh` 6.03 is not supported.

lsunlockhost

alias for `lsadmin limunlock`

DESCRIPTION

This command can only be used by LSF Administrators.

Provided for backward compatibility.

wggroup

modifies membership of Windows NT groups for an entire workgroup

SYNOPSIS

wggroup [**-r**] *user_name nt_group_name ...*

wggroup [**-h**]

DESCRIPTION

You must run this command on a host in a Windows NT workgroup. You should have administrative privileges on every host in the workgroup.

Modifies the specified Windows NT groups on every host in the workgroup that you have administrative privileges on.

By default, adds the specified user to the specified groups on each host, if that user is not already part of the group.

Use `-r` to remove the user from groups.

OPTIONS

`-r`

Removes the specified user accounts from the specified groups on each host, if the user belongs to the group.

`user_name`

Required. Specifies an existing user account to add or remove.

`nt_group_name ...`

Required. Specifies existing Windows NT groups to modify.

`-h`

Prints command usage to stderr and exits.

OUTPUT

For each host in the workgroup, returns the result of the operation (SUCCESS or FAILED).

wgpasswd

changes a user's password for an entire Windows NT workgroup

SYNOPSIS

wgpasswd [*user_name*]

wgpasswd [**-h**]

DESCRIPTION

You must run this command on a host in a Windows NT workgroup. You must have administrative privileges to change another user's password.

Prompts for old and new passwords, then changes the password on every host in the workgroup.

By default, modifies your own user account.

OPTIONS

user_name

Specifies the account to modify. You must have administrative privileges to change another user's password.

-h

Prints command usage to stderr and exits.

OUTPUT

For each host in the workgroup, returns the status of the operation (SUCCESS or FAILED).

FILES

Modifies the LSF password file.

wguser

modifies user accounts for an entire Windows NT workgroup

SYNOPSIS

wguser [**-r**] *user_name ...*

wguser [**-h**]

DESCRIPTION

You must run this command on a host in a Windows NT workgroup. You should have administrative privileges on every host in the workgroup.

Modifies accounts on every host in the workgroup that you have administrative privileges on.

By default, prompts for a default password to use for all of the accounts, and then creates the specified user accounts on each host, if they do not already exist.

Use `-r` to remove accounts from the workgroup.

OPTIONS

`-r`

Removes the specified user accounts from each host, if they exist.

`user_name ...`

Required. Specifies the accounts to add or remove.

`-h`

Prints command usage to stderr and exits.

OUTPUT

For each host in the workgroup, returns the result of the operation (SUCCESS or FAILED).

PART

II

Environment Variables

Environment Variables

BSUB_BLOCK

Description If set, tells NIOS that it is running in batch mode.

Default Undefined

Notes If you submit a job with the `-K` option of `bsub`, which is synchronous execution, then BSUB_BLOCK is set. Synchronous execution means you have to wait for the job to finish before you can continue.

Where Defined Set internally

Product Batch

See Also The `-K` option of `bsub`

BSUB_QUIET

Syntax **`BSUB_QUIET=`***`any_value`*

Description Controls the printing of information about job submissions. If set, `bsub` will not print any information about job submission. For example, it will not print <<Job is submitted to default queue <normal>>, nor <<Waiting for dispatch>>.

Default Undefined

Where Defined From the command line

Example `BSUB_QUIET=1`

Product Batch, JobScheduler

BSUB_QUIET2

Syntax **`BSUB_QUIET2=`***`any_value`*

Description Suppresses the printing of information about job completion when a job is submitted with the `bsub -K` option.

If set, `bsub` will not print information about job completion to `stdout`. For example, when this variable is set, the message <<Job is finished>> will not be written to `stdout`.

If BSUB_QUIET and BSUB_QUIET2 are both set, no job messages will be printed to `stdout`.

Default Undefined

Where Defined From the command line

Example `BSUB_QUIET2=1`

Product Batch, JobScheduler

BSUB_STDERR

Syntax **`BSUB_STDERR=y`**

Description Redirects LSF messages for `bsub` to `stderr`.

By default, when this parameter is not set, LSF messages for `bsub` are printed to `stdout.`

When this parameter is set, LSF messages for `bsub` are redirected to `stderr`.

Default Undefined

Where Defined From the command line on UNIX. For example, in csh:

```
setenv BSUB_STDERR Y
```

From the Control Panel on Windows NT, as an environment variable

Product Batch

CLEARCASE_ROOT

Syntax **`CLEARCASE_ROOT=`***`path`*

Description The path to the Rational ClearCase view.

Notes If you want to submit a batch job from a ClearCase view, then CLEARCASE_ROOT must be defined. You should submit these jobs with `csub` rather than `bsub`. `csub` is used only with Rational ClearCase.

For interactive jobs, set LSF_JOB_STARTER to the ClearCase job starter.

Where Defined In the Job Starter, or from the command line

Example

```
CLEARCASE_ROOT=/view/myview
```

Product Batch

See Also `ClearCase`, Job Starter, LSF_JOB_STARTER

ENV_LSF_ALARM_CONTEXT

Description Identifies the context in which an alarm occurs.

Notes Alarms are defined in `lsb.alarms`.

Where Defined Set internally when an alarm is triggered

Product JobScheduler

See Also `lsb.alarms`

ENV_LSF_ALARM_NAME

Description Identifies the alarm name.

Notes Alarms are defined in `lsb.alarms`.

Where Defined Set internally when an alarm is triggered

See Also `lsb.alarms`

ENV_LSF_ALARM_SEVERITY

Description Identifies the severity of the alarm.

Notes Alarms are defined in `lsb.alarms`.

Where Defined Set internally when an alarm is triggered

See Also `lsb.alarms`

ENV_LSF_ALARM_SOURCE

Description Identifies the source of the alarm.

Notes Alarms are defined in `lsb.alarms`.

Where Defined Set internally when an alarm is triggered

See Also `lsb.alarms`

KRB5CCNAME

Syntax **KRB5CCNAME=***file_name*

Description The path to the file containing the DCE credentials. Used when running in the DCE environment.

LM_LICENSE_FILE

Syntax **LM_LICENSE_FILE=***file_name*

Description The path to where the license file is found. The file name is the name of the license file.

Default `/usr/local/flexlm/licenses/license.dat`

Notes A FLEXlm variable read by the `lmgrd` daemon.

Where Defined From the command line

See Also See "lsf.conf" under "LSF_LICENSE_FILE" on page 420

LS_EXEC_T

Syntax **LS_EXEC_T= START | END | CHKPNT | JOB_CONTROLS**

Description Indicates execution type for a job. LS_EXEC_T is set to:

- START or END for a job when the job begins executing or when it completes execution
- CHKPNT when the job is checkpointed
- JOB_CONTROLS when a control action is initiated

Where Defined Set by SBD during job execution

LS_JOBPID

Description The process ID of the job.

Product Batch

Where Defined During job execution, SBD sets LS_JOBPID to be the same as the process ID assigned by the operating system.

LS_SUBCWD

Description The current working directory (`cwd`) of the submission host where the remote task command was executed.

The way this parameter is set by LSF is as follows:

1. LSF looks for the PWD environment variable. If it finds it, sets LS_SUBCWD to PWD.
2. If the PWD environment variable does not exist, LSF looks for the CWD environment variable. If it finds CWD, sets LS_SUBCWD to CWD.
3. If the CWD environment variable does not exist, LSF calls the `getwd()` system function to retrieve the current working directory pathname. LSF sets LS_SUBCWD to the value that is returned.

Where Defined Set by SBD

Product Batch

LSB_CHKPNT_DIR

Syntax **`LSB_CHKPNT_DIR=`***`checkpoint_dir/job_ID`*

Description The directory containing files related to the submitted checkpointable job.

Valid Values: The value of *checkpoint_dir* is the directory you specified through the -k option of `bsub` when submitting the checkpointable job.

The value of *job_ID* is the job ID of the checkpointable job.

Where Defined: Set by LSF, based on the directory you specified when submitting a checkpointable job with the `-k` option of `bsub`.

Product: Batch, checkpointing

LSB_DEBUG

This parameter can be set from the command line or from `lsf.conf`. See "lsf.conf" under "LSB_DEBUG" on page 382.

LSB_DEBUG_CMD

This parameter can be set from the command line or from `lsf.conf`. See "lsf.conf" under "LSB_DEBUG_CMD" on page 383.

LSB_DEBUG_MBD

This parameter can be set from the command line with `badmin mbddebug` or from `lsf.conf`.

See "lsf.conf" under "LSB_DEBUG_MBD" on page 384.

LSB_DEBUG_NQS

This parameter can be set from the command line or from `lsf.conf`. See "lsf.conf" under "LSB_DEBUG_NQS" on page 385.

LSB_DEBUG_SBD

This parameter can be set from the command line with `badmin sbddebug` or from `lsf.conf`.

See "lsf.conf" under "LSB_DEBUG_SBD" on page 386.

LSB_DEFAULTPROJECT

Syntax **LSB_DEFAULTPROJECT=***project_name*

Description The name of the project to which resources consumed by a job will be charged.

Default Undefined

Notes If the LSF administrator defines a default project in the `lsb.params` configuration file, the system uses this as the default project. You can change the default project by setting LSB_DEFAULTPROJECT or by specifying a project name with the `-P` option of `bsub`.

If you submit a job without the `-P` option of `bsub`, but you defined LSB_DEFAULTPROJECT, then the job belongs to the project specified in LSB_DEFAULTPROJECT.

If you submit a job with the `-P` option of `bsub`, the job belongs to the project specified through the `-P` option.

Where Defined From the command line, or through the `-P` option of `bsub`

Example
```
LSB_DEFAULTPROJECT=engineering
```

Product Batch

See Also The DEFAULT_PROJECT parameter in `lsb.params`, the `-P` option of `bsub`

LSB_DEFAULTQUEUE

Syntax **LSB_DEFAULTQUEUE=***queue_name*

Description Defines the default queue of the Batch system.

Default MBD decides which is the default queue. You can override the default by defining LSB_DEFAULTQUEUE.

Notes If the LSF administrator defines a default queue in the `lsb.params` configuration file, then the system uses this as the default queue. Provided you have permission, you can change the default queue by setting LSB_DEFAULTQUEUE to a valid queue (see `bqueues` for a list of valid queues).

Where Defined From the command line

Product Batch

See Also The DEFAULT_QUEUE parameter in `lsb.params`.

LSB_ECHKPNT_METHOD

This parameter can be set as an environment variable and/or in `lsf.conf`. See "lsf.conf" under "LSB_ECHKPNT_METHOD" on page 387.

LSB_ECHKPNT_METHOD_DIR

This parameter can be set as an environment variable and/or in `lsf.conf`. See "lsf.conf" under "LSB_ECHKPNT_METHOD_DIR" on page 388.

LSB_ECHKPNT_KEEP_OUTPUT

This parameter can be set as an environment variable and/or in `lsf.conf`. See "lsf.conf" under "LSB_ECHKPNT_KEEP_OUTPUT" on page 388.

LSB_ERESTART_USRCMD

Syntax **`LSB_ERESTART_USRCMD=`***`command`*

Description Original command used to start the job.

This environment variable is set by `erestart` to pass the job's original start command to a custom `erestart` method `erestart.`*`method_name`*. The value of this variable is extracted from the job file of the checkpointed job.

If a job starter is defined for the queue to which the job was submitted, the job starter is also included in LSB_ERESTART_USRCMD. For example, if the job starter is `/bin/sh -c "%USRCMD"` in `lsb.queues`, and the job name is `myapp -d`, LSB_ERESTART_USRCMD will be set to:

```
/bin/sh -c "myapp -d"
```

Where Defined Set by `erestart` as an environment variable before a job is restarted

Product Batch

See Also LSB_ECHKPNT_METHOD, erestart, echkpnt

LSB_EVENT_ATTRIB

Syntax **`LSB_EVENT_ATTRIB="`***`event_name1 attribute1 event_name2 attribute2...`***`"`**

Description Sets the attributes of external events that are specified in the dependency condition of `bsub` when the job is submitted.

Where Defined Set to the external event attributes that are specified in the dependency condition of `bsub` when the job is submitted.

Product JobScheduler

See Also The dependency condition (`-w`) of `bsub`

LSB_EXECHOSTS

Description A list of hosts on which a batch job will run.

Where Defined Set by SBD

Product MultiCluster

LSB_EXIT_PRE_ABORT

Description The queue-level or job-level *pre_exec_command* can exit with this value if the job is to be aborted instead of being requeued or executed

Where Defined Set by SBD

Product Batch

See Also `lsb.queues`, or the `-E` option of `bsub`

LSB_EXIT_REQUEUE

Syntax **`LSB_EXIT_REQUEUE="`***exit_value1 exit_value2...*`"`**

Description Contains a list of exit values found in the queue's REQUEUE_EXIT_VALUES parameter defined in `lsb.queues`.

Valid Values Any positive integers

Default Undefined

Notes If LSB_EXIT_REQUEUE is defined, a job will be requeued if it exits with one of the specified values.

LSB_EXIT_REQUEUE is undefined if the parameter REQUEUE_EXIT_VALUES is undefined.

Where Defined Set by the system based on the value of the parameter REQUEUE_EXIT_VALUES in `lsb.queues`

Example
```
LSB_EXIT_REQUEUE="7 31"
```

Product Batch

See Also REQUEUE_EXIT_VALUES

LSB_FRAMES

Syntax **LSB_FRAMES=***start_number,end_number,step*

Description Determines the number of frames to be processed by a frame job.

Valid Values The values of *start_number*, *end_number*, and *step* are positive integers. Use commas to separate the values.

Default Undefined

Notes When the job is running, LSB_FRAMES will be set to the relative frames with the format LSB_FRAMES=*start_number,end_number,step*.

From the *start_number*, *end_number*, and *step*, the frame job can know how many frames it will process.

Where Defined Set by SBD

Example `LSB_FRAMES=10,20,1`

Product Batch

LSB_HOSTS

Syntax **LSB_HOSTS="***host_name...***"**

Description A list of hosts selected by LSF Batch to run the batch job.

Notes If a job is run on a single processor, the system sets LSB_HOSTS to the name of the host used. For parallel jobs, the system sets LSB_HOSTS to the names of all the hosts used.

Where Defined Set by SBD when the job is submitted. LSB_HOSTS is set only when the list of host names is less than 4096 bytes.

Product Batch

See Also LSB_MCPU_HOSTS

LSB_INTERACTIVE

Syntax **LSB_INTERACTIVE=Y**

Description Indicates an interactive job. When you submit an interactive job using `bsub -I`, the system sets LSB_INTERACTIVE to Y.

Valid Values LSB_INTERACTIVE=Y (if the job is interactive)

Default Undefined (if the job is not interactive)

Where Defined Set by SBD

Product Batch

LSB_JOB_STARTER

Syntax **LSB_JOB_STARTER=***binary*

Description Specifies an executable program that has the actual job as an argument.

Default Undefined

Notes
- Interactive Jobs

  If you want to run an interactive job that requires some preliminary setup, LSF provides a job starter function at the command level. A command-level job starter allows you to specify an executable file that will run prior to the actual job, doing any necessary setup and running the job when the setup is complete.

  If the environment variable LSB_JOB_STARTER is properly defined, SBD will invoke the job starter (rather than the job itself), supplying your commands as arguments.
- Batch Jobs

A job starter can also be defined at the queue level using the JOB_STARTER parameter, although this can only be done by the LSF administrator.

Where Defined From the command line

See Also The JOB_STARTER parameter in `lsb.queues`

Example

- UNIX

  The job starter is invoked from within a Bourne shell, making the command-line equivalent:

  `/bin/sh -c "$LSB_JOB_STARTER command [argument...]"`

  where `command [argument...]` are the command line arguments you specified in `lsrun`, `lsgrun`, or `ch`.

  If you define LSB_JOB_STARTER as follows:

  ```
  % setenv LSB_JOB_STARTER "/bin/csh -c"
  ```

  and run a simple C-shell job:

  ```
  % lsrun "'a.out; echo hi'"
  ```

  then the following will be invoked to correctly start the job:

  ```
  /bin/sh -c "/bin/csh -c 'a.out; echo hi'"
  ```

- Windows NT

  RES runs the job starter, passing it your commands as arguments:

  `LSB_JOB_STARTER command [argument...]`

  If you define LSB_JOB_STARTER as follows:

  ```
  set LSB_JOB_STARTER=C:\cmd.exe /C
  ```

  and run a simple DOS shell job:

  ```
  C:\> lsrun dir /p
  ```

  then the following will be invoked to correctly start the job:

  ```
  C:\cmd.exe /C dir /p
  ```

Product Batch

See Also The JOB_STARTER parameter in `lsb.queues`

LSB_JOBEXIT_STAT

Syntax **LSB_JOBEXIT_STAT=***exit_status*

Description Indicates a job's exit status.

Applies to post-execution commands. Post-execution commands are set with POST_EXEC in `lsb.queues`.

When the post-execution command is run, the environment variable LSB_JOBEXIT_STAT is set to the exit status of the job. Refer to the man page for the `wait`(2) command for the format of this exit status.

The post-execution command is also run if a job is requeued because the job's execution environment fails to be set up, or if the job exits with one of the queue's REQUEUE_EXIT_VALUES. The LSB_JOBPEND environment variable is set if the job is requeued. If the job's execution environment could not be set up, LSB_JOBEXIT_STAT is set to 0.

Valid Values Any positive integer

Where Defined Set by SBD

Product Batch

LSB_JOBFILENAME

Syntax **LSB_JOBFILENAME=***file_name*

Description The path to the batch job file.

Notes Specifies the path to the batch executable job file that invokes the batch job. The batch executable job file is a /bin/sh script on UNIX systems or a .BAT command script on Windows NT systems.

LSF Batch sets LSB_JOBFILENAME.

Product Batch

LSB_JOBID

Syntax **LSB_JOBID=***job_ID*

Description The job ID assigned by the Batch system. This is the ID of the job assigned by LSF, as shown by `bjobs`.

Valid Values Any positive integer

Where Defined Set by SBD, defined by MBD

Product Batch

See Also LSB_REMOTEJID

LSB_JOBINDEX

Syntax **LSB_JOBINDEX=***index*

Description Contains the job array index.

Valid Values Any integer greater than zero but less than the maximum job array size.

Notes LSB_JOBINDEX is set when each job array element is dispatched. Its value corresponds to the job array index. This variable is set only if the job is an element of a job array.

Where Defined Set during job execution based on `bsub` options.

Example You can use LSB_JOBINDEX in a shell script to select the job command to be performed based on the job array index.

For example:

```
if [$LSB_JOBINDEX -eq 1]; then
cmd1
fi
if [$LSB_JOBINDEX -eq 2]; then
cmd2
fi
```

Product Batch

See Also LSB_JOBINDEX_STEP, LSB_REMOTEINDEX

LSB_JOBINDEX_STEP

Syntax **LSB_JOBINDEX_STEP=***step*

Description Step at which single elements of the job array are defined.

Valid Values Any integer greater than zero but less than the maximum job array size

Default 1

Notes LSB_JOBINDEX_STEP is set when a job array is dispatched. Its value corresponds to the step of the job array index. This variable is set only for job arrays.

Where Defined Set during job execution based on `bsub` options.

Example The following is an example of an array where a step of 2 is used:

```
array[1-10:2]
elements:1 3 5 7 9
```

If this job array is dispatched, then LSB_JOBINDEX_STEP=2

Product Batch

See Also LSB_JOBINDEX

LSB_JOBNAME

Syntax **LSB_JOBNAME=***job_name*

Description The name of the job defined by the user at submission time.

Default The job's command line

Notes The name of a job can be specified explicitly when you submit a job. The name does not have to be unique. If you do not specify a job name, the job name defaults to the actual batch command as specified on the `bsub` command line.

Where Defined Set by SBD

Example When you submit a job using the `-J` option of `bsub`, for example:

```
% bsub -J "myjob" job
```

SBD sets LSB_JOBNAME to the job name that you specified:

```
LSB_JOBNAME=myjob
```

Product Batch

LSB_JOBPEND

Description Set if the job is requeued.

Where Defined Set by SBD

Product Batch

See Also LSB_JOBEXIT_STAT, REQUEUE_EXIT_VALUES

LSB_JOBPGIDS

Description A list of the current process group IDs of the job.

Where Defined The process group IDs are assigned by the operating system, and LSB_JOBPGIDS is set by SBD.

Product Batch

See Also LSB_JOBPIDS

LSB_JOBPIDS

Description A list of the current process IDs of the job.

Where Defined The process IDs are assigned by the operating system, and LSB_JOBPIDS is set by SBD.

Product Batch

See Also LSB_JOBPGIDS

LSB_MAILSIZE

Syntax **LSB_MAILSIZE=***value*

Description Gives an estimate of the size of the batch job output when the output is sent by email. It is not necessary to configure LSB_MAILSIZE_LIMIT.

LSF sets LSB_MAILSIZE to the size in KB of the job output, allowing the custom mail program to intercept output that is larger than desired.

LSB_MAILSIZE is not recognized by the LSF default mail program. To prevent large job output files from interfering with your mail system, use LSB_MAILSIZE_LIMIT to explicitly set the maximum size in KB of the email containing the job information.

Valid Values

- A positive integer

If the output is being sent by email, LSB_MAILSIZE is set to the estimated mail size in kilobytes.

- -1

  If the output fails or cannot be read, LSB_MAILSIZE is set to -1 and the output is sent by email using LSB_MAILPROG if specified in `lsf.conf`.

- Undefined

  If you use the `-o` or `-e` options of `bsub`, the output is redirected to an output file. Because the output is not sent by email in this case, LSB_MAILSIZE is not used and LSB_MAILPROG is not called.

  If the `-N` option is used with the `-o` option of `bsub`, LSB_MAILSIZE is not set.

Where Defined Set by SBD when the custom mail program specified by LSB_MAILPROG in `lsf.conf` is called.

LSB_MCPU_HOSTS

Syntax **`LSB_MCPU_HOSTS="`***`host_nameA num_processors1 host_nameB num_processors2...`***`"`**

Description Contains a list of the hosts and the number of CPUs used to run a job.

Valid Values *`num_processors1`*, *`num_processors2`*,... refer to the number of CPUs used on *`host_nameA`*, *`host_nameB`*,..., respectively

Default Undefined

Notes The environment variables LSB_HOSTS and LSB_MCPU_HOSTS both contain the same information, but the information is presented in different formats. If you look at the usage example, you see that LSB_MCPU_HOSTS uses a shorter format than LSB_HOSTS. As a general rule, SBD sets both these variables. However, for some parallel jobs, LSB_HOSTS is not set.

For parallel jobs, several CPUs are used, and the length of LSB_HOSTS can become very long. SBD needs to spend a lot of time parsing the string. If the size of LSB_HOSTS exceeds 4096 bytes, LSB_HOSTS is ignored, and SBD sets only LSB_MCPU_HOSTS.

If you want to verify the hosts and CPUs used for your dispatched job, check the value of LSB_HOSTS for single CPU jobs, and check the value of LSB_MCPU_HOSTS for parallel jobs.

Where Defined Set by SBD at job submission

Example When the you submit a job with the -m and -n options of `bsub`, for example,

```
% bsub -m "hostA hostB" -n 6 job
```

SBD sets the environment variables LSB_HOSTS and LSB_MCPU_HOSTS as follows:

```
LSB_HOSTS= "hostA hostA hostA hostB hostB hostB"
```

```
LSB_MCPU_HOSTS="hostA 3 hostB 3"
```

Both variables are set in order to maintain compatibility with older versions.

Product Batch, Parallel

See Also LSB_HOSTS

LSB_NQS_PORT

This parameter can alternatively be defined in `lsf.conf` or in the services database such as `/etc/services`.

See “lsf.conf” under “LSB_NQS_PORT” on page 400 for more details.

LSB_OLD_JOBID

Syntax **LSB_OLD_JOBID=***job_ID*

Description The job ID of a job at the time it was checkpointed.

When a job is restarted, it is assigned a new job ID and LSB_JOBID is replaced with the new job ID. LSB_OLD_JOBID identifies the original ID of a job before it is restarted.

Valid Values Any positive integer

Where Defined Set by SBD, defined by MBD

Product Batch

See Also LSB_JOBID

LSB_OUTPUT_TARGETFAILED

Syntax **`LSB_OUTPUT_TARGETFAILED=Y`**

Description Indicates that LSF cannot access the output file specified for a job submitted the `bsub -o` option.

Valid Values Set to Y if the output file cannot be accessed; otherwise, it is undefined.

Where Defined Set by SBD during job execution

Product Batch

LSB_QUEUE

Description The name of the queue from which the job is dispatched.

Where Defined Set by SBD

Product Batch

LSB_REMOTEINDEX

Syntax **LSB_REMOTEINDEX=***index*

Description The job array index of a remote MultiCluster job. LSB_REMOTEINDEX is set only if the job is an element of a job array.

Valid Values Any integer greater than zero, but less than the maximum job array size

Where Defined Set by SBD

Product Batch

See Also LSB_JOBINDEX, "MAX_JOB_ARRAY_SIZE" on page 299 in "lsb.params"

LSB_REMOTEJID

Syntax **LSB_REMOTEJID=***job_ID*

Description The job ID of a remote MultiCluster job.

Where Defined Set by SBD, defined by MBD

Product Batch

See Also LSB_JOBID

LSB_RESTART

Syntax **LSB_RESTART=Y**

Description Indicates that a job has been restarted or migrated.

Valid Values Set to Y if the job has been restarted or migrated; otherwise, it is undefined.

Notes: If a batch job is submitted with the `-r` option of `bsub`, and is restarted because of host failure, then LSB_RESTART is set to Y. If a checkpointable job is submitted with the `-k` option of `bsub`, then LSB_RESTART is set to Y when the job is restarted. If `bmig` is used to migrate a job, then LSB_RESTART is set to Y when the migrated job is restarted.

If the job is not a restarted job, then LSB_RESTART is not set.

Where Defined: Set by SBD during job execution

Product: Batch

See Also: LSB_RESTART_PGID, LSB_RESTART_PID

LSB_RESTART_PGID

Syntax: **LSB_RESTART_PGID=***pgid*

Description: The process group ID of the checkpointed job when the job is restarted.

Notes: When a checkpointed job is restarted, the operating system assigns a new group process ID to the job. Batch sets LSB_RESTART_PGID to the new group process ID.

Where Defined: Set by Batch during restart of a checkpointed job

Product: Batch, Checkpointing

See Also: LSB_RESTART_PID, LSB_RESTART

LSB_RESTART_PID

Syntax: **LSB_RESTART_PID=***pid*

Description: The process ID of the checkpointed job when the job is restarted.

Notes When a checkpointed job is restarted, the operating system assigns a new process ID to the job. Batch sets LSB_RESTART_PID to the new process ID.

Where Defined Defined by Batch during restart of a checkpointed job

Product Batch, Checkpointing

See Also LSB_RESTART_PGID, LSB_RESTART

LSB_SUSP_REASONS

Syntax **LSB_SUSP_REASONS=***integer*

Description An integer representing suspend reasons. Suspend reasons are defined in `lsbatch.h`.

This parameter is set when a job goes to system-suspended (SSUSP) or user-suspended status (USUSP). It indicates the exact reason why the job was suspended.

To determine the exact reason, you can test the value of LSB_SUSP_REASONS against the symbols defined in `lsbatch.h`.

Default Undefined

Where Defined Set by SBD

Product Batch

See Also LSB_SUSP_SUBREASONS

LSB_SUSP_SUBREASONS

Syntax **LSB_SUSP_SUBREASONS=***integer*

Description An integer representing load indices. Load index values are defined in `lsf.h` as:

Load Index	Value
R15S	0
R1M	1
R15M	2
UT	3
PG	4
IO	5
LS	6
IT	7
TMP	8
SWP	9
MEM	10

When a value of the symbol SUSP_LOAD_REASON is set in LSB_SUSP_REASONS, it means the job is suspended by load, and LSB_SUSP_SUBREASONS set to one of the load index values.

You can use LSB_SUSP_SUBREASONS to determine which load index caused the job to be suspended.

LSB_SUSP_REASONS and LSB_SUSP_SUBREASONS can be used together in job control to determine the exact load threshold that caused a job to be suspended.

Default Undefined

Where Defined Set by SBD

Product Batch

See Also LSB_SUSP_REASONS

LSF_CMD_LOGDIR

This parameter can be set from the command line or from `lsf.conf`.

See "lsf.conf" under "LSF_CMD_LOGDIR" on page 409.

LSF_DEBUG_CMD

This parameter can be set from the command line or from `lsf.conf`.

See "lsf.conf" under "LSB_DEBUG_MBD" on page 384.

LSF_DEBUG_LIM

This parameter can be set from the command line or from `lsf.conf`.

See "lsf.conf" under "LSF_DEBUG_LIM" on page 411.

LSF_DEBUG_RES

This parameter can be set from the command line or from `lsf.conf`.

See "lsf.conf" under "LSF_DEBUG_RES" on page 412.

LSF_EAUTH_AUX_DATA

Syntax **`LSF_EAUTH_AUX_DATA=`***`path/file_name`*

Description The full path to the temporary file on the local filesystem that is used for storing auxiliary authentication information.

Notes Credentials are passed between invocations of `eauth` and the daemons through the file defined by LSF_EAUTH_AUX_DATA.

To allow daemons to call `eauth` to authenticate each other, you must define LSF_AUTH_DAEMONS.

Where Defined Set internally by `eauth`

See Also LSF_AUTH_DAEMONS, LSF_EAUTH_AUX_PASS

LSF_EAUTH_AUX_PASS

Syntax **LSF_EAUTH_AUX_PASS=yes**

Description Grants permission.

LSF_EAUTH_AUX_PASS is passed to `eauth -c` when LSF_EAUTH_CLIENT=user, and it tells `eauth` that it has permission to forward auxiliary authentication data.

To allow daemons to call `eauth` to authenticate each other, you must define LSF_AUTH_DAEMONS.

Where Defined Set internally

Product SUN HPC

See Also LSF_EAUTH_AUX_DATA, LSF_EAUTH_CLIENT

LSF_EAUTH_CLIENT

Syntax

- SUN HPC
  LSF_EAUTH_CLIENT=mbatchd | sbatchd | pam | res | user
- LSF3.2+
  LSF_EAUTH_CLIENT=user

Description A string that specifies the daemon or user that is calling `eauth -c`.

Notes Sets the context for the call to `eauth`, and allows the `eauth` writer to perform daemon authentication.

Where Defined Set internally by the LSF libraries, or by the daemon calling `eauth -c`.

Product LSF3.2+

See Also LSF_EAUTH_SERVER

LSF_EAUTH_SERVER

Syntax

- SUN HPC

  LSF_EAUTH_SERVER=mbatchd | sbatchd | pam | res

- LSF3.2+

LSF_EAUTH_SERVER=mbatchd | res

Description Specifies the daemon or user that is calling `eauth -s`

Notes Sets the context for the call to `eauth`, and allows the eauth writer to perform daemon authentication.

Where Defined Set internally by the LSF libraries, or by the daemon calling `eauth -s`

Product LSF3.2+

See Also LSF_EAUTH_CLIENT

LSF_EAUTH_UID

Syntax **LSF_EAUTH_UID=***user_ID*

Description Specifies the user ID under which `eauth -s` must run.

Where Defined Set by the LSF daemon which executes `eauth`.

Product Batch

See Also See “lsf.sudoers” under “LSF_EAUTH_USER” on page 459.

LSF_INTERACTIVE_STDERR

This parameter can alternatively be defined in `lsf.conf`.

See “lsf.conf” under “LSF_INTERACTIVE_STDERR” on page 418 for more details.

LSF_JOB_STARTER

Syntax **LSF_JOB_STARTER=***binary*

Description Specifies an executable program that has the actual job as an argument.

Default Undefined

Notes Interactive Jobs

If you want to run an interactive job that requires some preliminary setup, LSF provides a job starter function at the command level. A command-level job starter allows you to specify an executable file that will run prior to the actual job, doing any necessary setup and running the job when the setup is complete.

If LSF_JOB_STARTER is properly defined, RES will invoke the job starter (rather than the job itself), supplying your commands as arguments.

Batch Jobs

A Job Starter can also be defined at the queue level using the JOB_STARTER parameter, although this can only be done by the LSF administrator.

Where Defined From the command line

Example

- UNIX

  The job starter is invoked from within a Bourne shell, making the command-line equivalent:

  ```
  /bin/sh -c "$LSF_JOB_STARTER command [argument...]"
  ```

  where *command* [*argument...*] are the command line arguments you specified in `lsrun`, `lsgrun`, or `ch`.

  If you define LSF_JOB_STARTER as follows:

  ```
  % setenv LSF_JOB_STARTER "/bin/csh -c"
  ```

  and run a simple C-shell job:

  ```
  % lsrun "'a.out; echo hi'"
  ```

The following will be invoked to correctly start the job:

```
/bin/sh -c "/bin/csh -c 'a.out; echo hi'"
```

- Windows NT

  RES runs the job starter, passing it your commands as arguments:

  `LSF_JOB_STARTER` *`command`* `[`*`argument...`*`]`

  If you define LSF_JOB_STARTER as follows:

  ```
  set LSF_JOB_STARTER=C:\cmd.exe /C
  ```

  and run a simple DOS shell job:

  ```
  C:\> lsrun dir /p
  ```

  then the following will be invoked to correctly start the job:

  ```
  C:\cmd.exe /C dir /p
  ```

See Also The JOB_STARTER parameter in `lsb.queues`

LSF_LIM_DEBUG

This parameter can be set from the command line or from `lsf.conf`.

See "lsf.conf" under "LSF_LIM_DEBUG" on page 421.

LSF_LOGDIR

This parameter can be set from the command line or from `lsf.conf`.

See "lsf.conf" under "LSF_LOGDIR" on page 424.

LSF_MASTER

Description Specifies whether ELIM has been started on the master host.

Notes LIM communicates with ELIM through two environment variables: LSF_MASTER and LSF_RESOURCES.

LSF_MASTER is set to `Y` when LIM starts ELIM on the master host. It is set to N or is undefined otherwise.

LSF_MASTER can be used to test whether the ELIM should report on cluster-wide resources that only need to be collected on the master host.

When Defined Set by LIM when ELIM is started

See Also LSF_RESOURCES

LSF_NIOS_DEBUG

This parameter can be set from the command line or from `lsf.conf`. See “lsf.conf” under “LSF_NIOS_DEBUG” on page 427.

LSF_NIOS_DIE_CMD

Syntax **LSF_NIOS_DIE_CMD=***command*

Description If set, the command defined by LSF_NIOS_DIE_CMD is executed before NIOS exits.

Default Undefined

Where Defined From the command line

LSF_NIOS_IGNORE_SIGWINDOW

Syntax **LSF_NIOS_IGNORE_SIGWINDOW=***any_value*

Description If defined, the NIOS will ignore the SIGWINDOW signal.

Default Undefined

Notes When the signal SIGWINDOW is defined, some tasks appear to die when they receive the SIGWINDOW while doing I/O. By defining LSF_NIOS_IGNORE_SIGWINDOW, these tasks are given the chance to ignore the signal.

Where Defined From the command line

LSF_NIOS_PEND_TIMEOUT

Syntax **LSF_NIOS_PEND_TIMEOUT=** *minutes*

Description Applies only to interactive batch jobs.

Maximum amount of time that an interactive batch job can remain pending.

If this parameter is defined, and an interactive batch job is pending for longer than the specified time, the interactive batch job is terminated.

Valid Values Any integer greater than zero

Default Undefined

Product LSF Batch

LSF_RESOURCES

Syntax **LSF_RESOURCES=***dynamic_shared_resource_name...*

Description Space-separated list of customized dynamic shared resources that the ELIM is responsible for collecting.

Valid Values A resource name is only put in the list if the host on which the ELIM is running shares an instance of that resource.

Notes LIM communicates with the ELIM through two environment variables: LSF_MASTER and LSF_RESOURCES.

LSF_MASTER is set to `Y` when LIM starts ELIM on the master host. It is set to N or is undefined otherwise.

LSF_RESOURCES is set to a space-separated string of dynamic shared resources for which the ELIM on that host is responsible for collecting. LSF_RESOURCES gets passed to ELIM from LIM.

When Defined By LIM when ELIM is invoked

Example `LSF_RESOURCES="resource1 resource2 resource3"`

See Also LSF_MASTER

LSF_USE_HOSTEQUIV

Description Used for authentication purposes. If LSF_USE_HOSTEQUIV is defined, LSF will trust all hosts configured in the LSF cluster that are defined in `hosts.equiv`, or in `.rhosts` in the user's home directory.

Default Undefined

Notes If LSF_USE_HOSTEQUIV is not defined, all normal users in the cluster can execute remote jobs on any host. If LSF_ROOT_REX is set, `root` can also execute remote jobs with the same permission test as for normal users.

See Also "LSF_ROOT_REX" on page 435, "LSF_AUTH" on page 408 in "lsf.conf"

LSF_USER_DOMAIN

Syntax LSF_USER_DOMAIN = *domain_name* | `.`

Description Set during LSF installation or setup. If you modify this parameter in an existing cluster, you probably have to modify passwords and configuration files also.

Windows or mixed UNIX-Windows clusters only.

Enables default user mapping, and specifies the LSF user domain. The period (`.`) specifies local accounts, not domain accounts.

- a user name specified without a domain is intrepreted (on a Windows host) as belonging to the LSF user domain
- a user name specified with the domain name of the LSF user domain is invalid
- in a mixed cluster, this parameter defines a 2-way, 1:1 user map between UNIX user accounts and Windows user accounts belonging to the specified domain, as long as the accounts have the

same user name. This means jobs submitted by the Windows user account can run on a UNIX host, and jobs submitted by the UNIX account can run on any Windows host that is available to the Windows user account.

If this parameter is undefined, the default user mapping is not enabled. You can still configure user mapping at the user or system level. User account mapping is required to run cross-platform jobs in a UNIX-Windows mixed cluster.

Where Defined `lsf.conf`

Default

- If you upgrade from LSF 4.0.1 or earlier, the default is the existing LSF user domain.
- For a new, Windows-only cluster, this parameter is undefined (no LSF user domain, no default user mapping).
- For a new, mixed UNIX-Windows cluster, the default is the domain that the Windows installation account belongs to. This can be modified during LSF installation.

.

PART

III

Configuration Files

hosts

Overview

The LSF `hosts` file is stored in LSF_CONFDIR. The format of `LSF_CONFDIR/hosts` is the same as for `/etc/hosts`.

For hosts with multiple IP addresses and different official host names configured at the system level, this file associates the host names and IP addresses in LSF. Hosts with only one IP address, or hosts with multiple IP addresses that already resolve to a unique official host name should not be configured in this file: they are resolved using the default method for your system.

hosts File Structure

One line for each IP address, consisting of the IP address, followed by the official host name, optionally followed by host aliases, all separated by spaces or tabs.

Use consecutive lines for IP addresses belonging to the same host. You can assign different aliases to different addresses.

Use a pound sign (#) to indicate a comment (the rest of the line is not read by LSF).

A call to `gethostbyname(3N)` returns a `hostent` structure containing the union of all addresses and aliases from each line containing a matching official host name or alias.

IP Address

Written using the conventional dotted decimal notation (`nnn.nnn.nnn.nnn`) and interpreted using the `inet_addr` routine from the Internet address manipulation library, `inet(3N)`.

Official Host Name

The official host name. Single character names are not allowed.

Specify -GATEWAY or -GW as part of the host name if the host serves as a GATEWAY.

Specify -TAC as the last part of the host name if the host is a TAC and is a DoD host.

Specify the host name in the format defined in Internet RFC 952 which states:

> A "name" (Net, Host, Gateway, or Domain name) is a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9), minus sign (-), and period (.). Periods are only allowed when they serve to delimit components of "domain style names". (See RFC 921, "Domain Name System Implementation Schedule", for background). No blank or space characters are permitted as part of a name. No distinction is made between upper and lower case. The first character must be an alpha character. The last character must not be a minus sign or a period.

RFC 952 has been modified by RFC 1123 to relax the restriction on the first character being a digit.

For maximum interoperability with the Internet, you should use host names no longer than 24 characters for the host portion (exclusive of the domain component).

Aliases

Optional. Aliases to the host name.

Example hosts File

```
192.168.1.1 hostA hostB
192.168.2.2 hostA hostC host-C
```

In this example, `hostA` has 2 IP addresses and 3 aliases. The alias `hostB` specifies the first address, and the aliases `hostC` and `host-C` specify the second address. LSF uses the official host name, `hostA`, to identify that both IP addresses belong to the same host.

lsb.hosts

Overview The `lsb.hosts` file contains host-related configuration information for the server hosts in the cluster. This file is optional. All sections are optional.

Contents

Host Section

Description

Optional. Defines the hosts, host types, and host models used as server hosts, and contains per-host configuration information. If this section is not configured, LSF uses all hosts in the cluster as server hosts.

Each host, host model or host type can be configured to:

- Limit the maximum number of jobs run in total
- Limit the maximum number of jobs run by each user
- Run jobs only under specific load conditions
- Run jobs only under specific time windows

The entries in a line for a host override the entries in a line for its model or type.

When you modify the cluster by adding or removing hosts, no changes are made to `lsb.hosts`. This does not affect the default configuration, but if hosts, host models, or host types are specified in this file, you should check this file whenever you make changes to the cluster and update it manually if necessary.

Host Section Structure

The first line consists of keywords identifying the load indices that you wish to configure on a per-host basis. The keyword HOST_NAME must be used; the others are optional. Load indices not listed on the keyword line do not affect scheduling decisions.

Each subsequent line describes the configuration information for one host, host model or host type. Each line must contain one entry for each keyword. Use empty parentheses ( ) or a dash (-) to specify the default value for an entry.

HOST_NAME

Required. Specify the name, model, or type of a host, or the keyword `default`.

host name The name of a host defined in `lsf.cluster.`*`cluster_name`*. The official host name returned by `gethostbyname(3)`.

host model A host model defined in `lsf.shared`.

host type A host type defined in `lsf.shared`.

default The reserved host name `default` indicates all hosts in the cluster not otherwise referenced in the section (by name or by listing its model or type).

CHKPNT

Description If C, checkpoint copy is enabled. With checkpoint copy, all opened files are automatically copied to the checkpoint directory by the operating system when a process is checkpointed.

Example

```
HOST_NAME  CHKPNT
hostA         C
```

Compatibility Checkpoint copy is only supported on Cray systems.

Default No checkpoint copy.

DISPATCH_WINDOW

Description The time windows in which jobs from this host, host model, or host type are dispatched. Once dispatched, jobs are no longer affected by the dispatch window.

Default Undefined (always open).

JL/U

Description Per-user job slot limit for the host. Maximum number of job slots that each user can use on this host.

Example

```
HOST_NAME  JL/U
hostA         2
```

Default Unlimited

MIG

Description Enables job migration and specifies the migration threshold, in minutes.

If a checkpointable or rerunnable job dispatched to the host is suspended (SSUSP state) for longer than the specified number of minutes, the job is migrated. A value of 0 specifies that a suspended job should be migrated immediately.

If a migration threshold is defined at both host and queue levels, the lower threshold is used.

Example

```
HOST_NAME   MIG
hostA        10
```

In this example, the migration threshold is 10 minutes.

Default Undefined (no migration)

MXJ

Description The number of job slots on the host.

By default, the number of running and suspended jobs on a host cannot exceed the number of job slots. If preemptive scheduling is used, the suspended jobs are not counted as using a job slot.

On multiprocessor hosts, to fully use the CPU resource, make the number of job slots equal to or greater than the number of processors.

Default Unlimited

load_index

Syntax

load_index
loadSched[/*loadStop*]

Specify `io`, `it`, `ls`, `mem`, `pg`, `r15s`, `r1m`, `r15m`, `swp`, `tmp`, `ut`, or a non-shared custom external load index as a column. Specify multiple columns to configure thresholds for multiple load indices.

Description

Scheduling and suspending thresholds for dynamic load indices supported by LIM, including external load indices.

Each load index column must contain either the default entry or two numbers separated by a slash '/', with no white space. The first number is the scheduling threshold for the load index; the second number is the suspending threshold.

Queue-level scheduling and suspending thresholds are defined in `lsb.queues`. If both files specify thresholds for an index, those that apply are the most restrictive ones.

Example

```
HOST_NAME    mem     swp
hostA        100/10  200/30
```

This example translates into a `loadSched` condition of

```
mem>=100 && swp>=200
```

and a `loadStop` condition of

```
mem < 10 || swp < 30
```

Default

Undefined

Example of a Host Section

```
Begin Host
HOST_NAME MXJ JL/U r1m     pg    DISPATCH_WINDOW
hostA     1   -    0.6/1.6 10/20 (5:19:00-1:8:30 20:00-8:30)
SUNSOL    1   -    0.5/2.5 -     23:00-8:00
default   2   1    0.6/1.6 20/40 ()
End Host
```

SUNSOL is a host type defined in `lsf.shared`. This example `Host` section configures one host and one host type explicitly and configures default values for all other load-sharing hosts.

`HostA` runs one batch job at a time. A job will only be started on `hostA` if the `r1m` index is below 0.6 and the `pg` index is below 10; the running job is stopped if the `r1m` index goes above 1.6 or the `pg` index goes above 20. `HostA` only accepts batch jobs from 19:00 on Friday evening until 8:30 Monday morning and overnight from 20:00 to 8:30 on all other days.

For hosts of type SUNSOL, the `pg` index does not have host-specific thresholds and such hosts are only available overnight from 23:00 to 8:00.

The entry with host name default applies to each of the other hosts in the LSF cluster. Each host can run up to two jobs at the same time, with at most one job from each user. These hosts are available to run jobs at all times. Jobs may be started if the `r1m` index is below 0.6 and the `pg` index is below 20, and a job from the lowest priority queue is suspended if `r1m` goes above 1.6 or `pg` goes above 40.

HostGroup Section

Description

Optional. Defines host groups.

The name of the host group can then be used in other host group, host partition, and queue definitions, as well as on the command line. Specifying the name of a host group has exactly the same effect as listing the names of all the hosts in the group.

Structure

Host groups are specified in the same format as user groups in `lsb.users`.

The first line consists of two mandatory keywords, GROUP_NAME and GROUP_MEMBER. Subsequent lines name a group and list its membership.

The total number of host groups cannot be more than MAX_GROUPS (see `lsbatch.h` for details).

GROUP_NAME

Description An alphanumeric string representing the name of the host group.

You cannot use the reserved name `all`, and group names must not conflict with host names.

GROUP_MEMBER

Description A space-separated list of host names or previously defined host group names, enclosed in parentheses.

The names of hosts and host groups can appear on multiple lines because hosts can belong to multiple groups. The reserved name `all` specifies all hosts in the cluster. Use an exclamation mark (!) to specify that the group membership should be retrieved via `egroup`. Use a tilde (~) to exclude specified hosts or host groups from the list.

Examples of HostGroup Sections

Example 1

```
Begin HostGroup
GROUP_NAME  GROUP_MEMBER
groupA      (hostA hostD)
groupB      (hostF groupA hostK)
groupC      (!)
End HostGroup
```

This example defines three host groups:

- `groupA` includes `hostsA` and `hostD`.
- `groupB` includes `hostsF` and `hostK`, along with all hosts in `groupA`.
- the group membership of `groupC` will be retrieved via `egroup`.

Example 2

```
Begin HostGroup
GROUP_NAME   GROUP_MEMBER
groupA       (all)
groupB       (groupA ~hostA ~hostB)
groupC       (hostX hostY hostZ)
groupD       (groupC ~hostX)
groupE       (all ~groupC ~hostB)
groupF       (hostF groupC hostK)
End HostGroup
```

This example defines the following host groups:

- `groupA` contains all hosts in the cluster.
- `groupB` contains all the hosts in the cluster except for `hostA` and `hostB`.
- `groupC` contains only `hostX`, `hostY`, and `hostZ`.
- `groupD` contains the hosts in `groupC` except for `hostX`. Note that `hostX` must be a member of host group `groupC` to be excluded from `groupD`.
- `groupE` contains all hosts in the cluster excluding the hosts in `groupC` and `hostB`.
- `groupF` contains `hostF`, `hostK`, and the 3 hosts in `groupC`.

HostPartition Section

Description

Optional; used with host partition fairshare scheduling. Defines a host partition, which defines a fairshare policy at the host level.

Configure multiple sections to define multiple partitions.

The members of a host partition form a host group with the same name as the host partition.

Limitations on Queue Configuration

- If you configure a host partition, you cannot configure fairshare at the queue level.
- If a queue uses a host that belongs to a host partition, it should not use any hosts that don't belong to that partition. All the hosts in the queue should belong to the same partition. Otherwise, you might notice unpredictable scheduling behavior:
  - Jobs in the queue sometimes may be dispatched to the host partition even though hosts not belonging to any host partition have a lighter load.
  - If some hosts belong to one host partition and some hosts belong to another, only the priorities of one host partition are used when dispatching a parallel job to hosts from more than one host partition.

Structure

Each host partition always consists of 3 lines, defining the name of the partition, the hosts included in the partition, and the user share assignments.

HPART_NAME

Syntax **HPART_NAME =** *partition_name*

Description Specifies the name of the partition.

HOSTS

Syntax **HOSTS =** [[**~**]*host_name* | [**~**]*host_group* | **all**]...

Description Specifies the hosts in the partition, in a space-separated list.

A host cannot belong to multiple partitions.

Hosts that are not included in any host partition are controlled by the FCFS scheduling policy instead of the fairshare scheduling policy.

Optionally, use the reserved host name `all` to configure a single partition that applies to all hosts in a cluster.

Optionally, use the not operator (~) to exclude hosts or host groups from the list of hosts in the host partition.

Example

```
HOSTS = all ~hostK ~hostM
```

The partition includes all the hosts in the cluster, except for hosts K and M.

USER_SHARES

Syntax **USER_SHARES = [***user***,** *number_shares***]**...

Description Specifies user share assignments

- Specify at least one user share assignment.
- Enclose each user share assignment in square brackets, as shown.
- Separate a list of multiple share assignments with a space between the square brackets.
- *user*

  Specify users who are also configured to use the host partition. You can assign the shares:

  - To a single user (specify *user_name*)
  - To users in a group, individually (specify *group_name*`@`) or collectively (specify *group_name*)

- To users not included in any other share assignment, individually (specify the keyword `default`) or collectively (specify the keyword `others`)

By default, when resources are assigned collectively to a group, the group members compete for the resources according to FCFS scheduling. You can use hierarchical fairshare to further divide the shares among the group members.

When resources are assigned to members of a group individually, the share assignment is recursive. Members of the group and of all subgroups always compete for the resources according to FCFS scheduling, regardless of hierarchical fairshare policies.

- *number_shares*

  Specify a positive integer representing the number of shares of the cluster resources assigned to the user.

  The number of shares assigned to each user is only meaningful when you compare it to the shares assigned to other users or to the total number of shares. The total number of shares is just the sum of all the shares assigned in each share assignment.

Example of a HostPartition Section

```
Begin HostPartition
HPART_NAME = Partition1
HOSTS = hostA hostB
USER_SHARES = [groupA@, 3] [groupB, 7] [default, 1]
End HostPartition
```

lsb.params

Overview

The `lsb.params` file defines general parameters used by LSF Batch. This file contains only one section, named Parameters. MBD uses `lsb.params` for initialization. The file is optional. If not present, the LSF-defined defaults are assumed.

Some of the parameters that can be defined in `lsb.params` control timing within the LSF Batch system. The default settings provide good throughput for long-running batch jobs while adding a minimum of processing overhead in the batch daemons.

Contents

Parameters Section

This section and all the keywords in this section are optional. If keywords are not present, LSF Batch assumes default values for the corresponding keywords. The valid keywords for this section are:

AUTOADJUST_AT_NUM_PEND

Syntax **AUTOADJUST_AT_NUM_PEND =** *integer*

Description Part of skip scheduling configuration. ENABLE_AUTOADJUST and AUTOADJUST_AT_PERCENT must also be specified.

Specify the minimum number of a user's pending jobs that LSF must consider for dispatch before skipping the rest of that user's jobs.

Default Undefined

AUTOADJUST_AT_PERCENT

Syntax **AUTOADJUST_AT_PERCENT =** *integer*

Description Part of skip scheduling configuration. ENABLE_AUTOADJUST and AUTOADJUST_AT_NUM_PEND must also be specified.

Use a number between 1 and 100 to specify the skip percentage.

Example For example, to specify that LSF should skip users when it cannot place at least 25% of their jobs, set:

```
AUTOADJUST_AT_PERCENT = 25
```

Default Undefined

CLEAN_PERIOD

Syntax **CLEAN_PERIOD =** *seconds*

Description For non-repetitive jobs, the amount of time that job records for jobs that have finished or have been killed are kept in MBD core memory after they have finished.

Users can still see all jobs after they have finished using the `bjobs` command. For jobs that finished more than CLEAN_PERIOD seconds ago, use the `bhist` command.

Default 3600 (1 hour).

CPU_TIME_FACTOR

Syntax **CPU_TIME_FACTOR =** *number*

Description Used only with fairshare scheduling. CPU time weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the cumulative CPU time used by a user's jobs.

Default 0.7

COMMITTED_RUN_TIME_FACTOR

Syntax **COMMITTED_RUN_TIME_FACTOR =** *number*

Description Used only with fairshare scheduling. Committed run time weighting factor.

In the calculation of a user's dynamic priority, this factor determines the relative importance of the committed run time in the calculation. If the `-W` option of `bsub` is not specified at job submission and a RUN_LIMIT has not been set for the queue, the committed run time is not considered.

Valid Values Any positive number between 0.0 and 1.0

Default 0.0

DEFAULT_HOST_SPEC

Syntax **DEFAULT_HOST_SPEC =** *host_name* | *host_model*

Description The default CPU time normalization host for the cluster.

The CPU factor of the specified host or host model will be used to normalize the CPU time limit of all jobs in the cluster, unless the CPU time normalization host is specified at the queue or job level.

Default Undefined

DEFAULT_PROJECT

Syntax **DEFAULT_PROJECT =** *project_name*

Description The name of the default project. Specify any string.

When you submit a job without specifying any project name, and the environment variable LSB_DEFAULTPROJECT is not set, LSF automatically assigns the job to this project.

Default `default`

DEFAULT_QUEUE

Syntax **DEFAULT_QUEUE =** *queue_name ...*

Description Space-separated list of candidate default queues (candidates must already be defined in `lsb.queues`).

When you submit a job to LSF without explicitly specifying a queue, and the environment variable LSB_DEFAULTQUEUE is not set, LSF puts the job in the first queue in this list that satisfies the job's specifications subject to other restrictions, such as requested hosts, queue status, etc.

Default Undefined. When a user submits a job to LSF without explicitly specifying a queue, and there are no candidate default queues defined (by this parameter or by the user's environment variable LSB_DEFAULTQUEUE), LSF automatically creates a new queue named `default`, using the default configuration, and submits the job to that queue.

DISABLE_UACCT_MAP

Syntax **DISABLE_UACCT_MAP = y | Y**

Description Specify y or Y to disable user-level account mapping.

Default Undefined

ENABLE_AUTOADJUST

Syntax **ENABLE_AUTOADJUST = y | Y**

Description Specify y or Y to enable skip scheduling. AUTOADJUST_AT_NUM_PEND and AUTOADJUST_AT_PERCENT must also be specified.

Default Undefined

ENABLE_HIST_RUN_TIME

Syntax **ENABLE_HIST_RUN_TIME = y | Y**

Description Used only with fairshare scheduling. If set, enables the use of historical run time in the calculation of fairshare scheduling priority.

Default Undefined

EVENT_UPDATE_INTERVAL

Syntax **EVENT_UPDATE_INTERVAL =** *time_in_seconds*

Description Time interval to synchronize LSB_SHAREDIR and LSB_LOCALDIR for duplicate event logging. Use this parameter if NFS traffic is too high and you want to reduce network traffic.

If this parameter is not defined and duplicate event logging is configured (parameter LSB_LOCALDIR is defined in `lsf.conf`) the LSB_LOCALDIR and LSB_SHAREDIR are synchronized:

- When the MBD of the primary master host is started
- When any event is recorded and the parent MBD signals the child MBD to synchronize the two directories
- Every 30 seconds

If EVENT_UPDATE_INTERVAL is defined and duplicate event logging is configured (parameter LSB_LOCALDIR is defined in `lsf.conf`), LSB_LOCALDIR and LSB_SHAREDIR are synchronized:

- When the MBD of the primary master host is started
- Every EVENT_UPDATE_INTERVAL

Valid Values 1 to INFINIT_INT

INFINIT_INT is defined in `lsf.h`

Default Undefined

See Also LSB_LOCALDIR in `lsf.conf`

HIST_HOURS

Syntax **HIST_HOURS =** *hours*

Description Used only with fairshare scheduling. Determines a rate of decay for cumulative CPU time and historical run time.

To calculate dynamic user priority, LSF scales the actual CPU time using a decay factor, so that 1 hour of recently-used time is equivalent to 0.1 hours after the specified number of hours has elapsed.

To calculate dynamic user priority with historical run time, LSF scales the accumulated run time of finished jobs using the same decay factor, so that 1 hour of recently-used time is equivalent to 0.1 hours after the specified number of hours has elapsed.

When HIST_HOURS=0, CPU time and historical run time do not affect the calculation of dynamic user priority.

Default 5

JOB_ACCEPT_INTERVAL

Syntax **`JOB_ACCEPT_INTERVAL =`** *integer*

Description The number of dispatch turns to wait after dispatching a job to a host, before dispatching a second job to the same host. By default, a dispatch turn lasts 60 seconds (MBD_SLEEP_TIME in `lsb.params`).

If 0 (zero), a host may accept more than one job in each job dispatching interval. By default, there is no limit to the total number of jobs that can run on a host, so if this parameter is set to 0, a very large number of jobs might be dispatched to a host all at once. You may notice performance problems if this occurs.

JOB_ACCEPT_INTERVAL set at the queue level (`lsb.queues`) overrides JOB_ACCEPT_INTERVAL set at the cluster level (`lsb.params`).

Default 1

JOB_ATTA_DIR

Syntax **JOB_ATTA_DIR =** *dir*

Description The directory in which MBD saves the attached data of messages

JOB_ATTA_DIR specifies a shared directory for data attached to a job with the `bpost`(1) command. Use JOB_ATTA_DIR if you use `bpost`(1) and `bread`(1)to transfer large data files between jobs and want to avoid using space in LSB_SHAREDDIR. By default, the `bread`(1) command reads attachment data from the JOB_ATTA_DIR directory.

JOB_ATTA_DIR should be shared by all hosts in the cluster, so that any potential LSF master host can reach it. Like LSB_SHAREDIR, the directory should be owned and writable by the primary LSF administrator. The directory must have at least 1 MB of free space.

The attached data will be stored under the directory in the format:

`JOB_ATTA_DIR/`*`timestamp`*`.`*`jobid`*`.msgs/msg$msgindex`

On UNIX, specify an absolute path. For example:

```
JOB_ATTA_DIR=/opt/share/lsf_work
```

On Windows NT, specify a UNC path or a path with a drive letter. For example:

```
JOB_ATTA_DIR=\\HostA\temp\lsf_workor
```

```
JOB_ATTA_DIR=D:\temp\lsf_work
```

After adding JOB_ATTA_DIR to `lsb.params`, use `badmin reconfig` to reconfigure your cluster.

JOB_ATTA_DIR can be any valid UNIX or Windows NT path up to a maximum length of 256 characters.

Default Undefined

If JOB_ATTA_DIR is not specified, job message attachments are saved in `LSB_SHAREDIR/info/`.

JOB_DEP_LAST_SUB

Description Used only with job dependency scheduling.

If set to 1, whenever dependency conditions use a job name that belongs to multiple jobs, LSF evaluates only the most recently submitted job.

Otherwise, all the jobs with the specified name must satisfy the dependency condition.

Default Undefined

JOB_PRIORITY_OVER_TIME

Syntax **JOB_PRIORITY_OVER_TIME=***increment*/*interval*

Description JOB_PRIORITY_OVER_TIME enables automatic job priority escalation when MAX_USER_PRIORITY is also defined.

Valid Values *increment*

Specifies the value used to increase job priority every *interval* minutes. Valid values are positive integers.

interval

Specifies the frequency, in minutes, to *increment* job priority. Valid values are positive integers.

Default Undefined

Example JOB_PRIORITY_OVER_TIME=3/20

Specifies that every 20 minute *interval* *increment* to job priority of pending jobs by 3.

See Also "MAX_USER_PRIORITY" on page 301

JOB_SPOOL_DIR

Syntax **JOB_SPOOL_DIR =** *dir*

Description Specifies the directory for buffering batch standard output and standard error for a job

When JOB_SPOOL_DIR is defined, the standard output and standard error for the job is buffered in the specified directory.

Except for `bsub -is` and `bsub -Zs`, if JOB_SPOOL_DIR is not accessible or does not exist, output is spooled to the default job output directory `.lsbatch`.

For `bsub -is` and `bsub -Zs`, JOB_SPOOL_DIR must be readable and writable by the job submission user, and it must be shared by the master host, the submission host, and the execution host. If the specified directory is not accessible or does not exist, `bsub -is` and `bsub -Zs` cannot write to the default directory and the job will fail.

As LSF runs jobs, it creates temporary directories and files under JOB_SPOOL_DIR. By default, LSF removes these directories and files after the job is finished. See `bsub(1)` for information about job submission options that specify the disposition of these files.

On UNIX, specify an absolute path. For example:

```
JOB_SPOOL_DIR=/home/share/lsf_spool
```

On Windows NT, specify a UNC path or a path with a drive letter. For example:

```
JOB_SPOOL_DIR=\\HostA\share\spooldir
```

or

```
JOB_SPOOL_DIR=D:\share\spooldir
```

In a mixed UNIX/Windows NT cluster, specify one path for the UNIX platform and one for the Windows NT platform. Separate the two paths by a pipe character (|):

```
JOB_SPOOL_DIR=/usr/share/lsf_spool | \\HostA\share\spooldir
```

JOB_SPOOL_DIR can be any valid UNIX or Windows NT path up to a maximum length of 256 characters. This maximum path length includes the temporary directories and files that LSF Batch creates as jobs run. The path you specify for JOB_SPOOL_DIR should be as short as possible to avoid exceeding this limit.

Default Undefined

Batch job output (standard output and standard error) is sent to the `.lsbatch` directory on the execution host:

- On UNIX: `$HOME/.lsbatch`
- On Windows NT: `%windir%\lsbtmp`*`user_id`*`\.lsbatch`

  If %HOME% is specified in the user environment, uses that directory instead of %windir% for spooled output.

JOB_TERMINATE_INTERVAL

Syntax **`JOB_TERMINATE_INTERVAL =`** *seconds*

Description UNIX only.

Specifies the time interval in seconds between sending SIGINT, SIGTERM, and SIGKILL when terminating a job. When a job is terminated, the job is sent SIGINT, SIGTERM, and SIGKILL in sequence with a sleep time of JOB_TERMINATE_INTERVAL between sending the signals. This allows the job to clean up if necessary.

Default 10

LSB_ACCT_AGE

Syntax **`LSB_ACCT_AGE=`** *days*

Description Enables automatic archiving of LSF accounting log files, and specifies the archive interval. LSF archives the current log file if the length of time from its creation date exceeds the specified number of days.

See Also LSB_ACCT_SIZE also enables automatic archiving.

LSB_ACCT_MAX_FILES enables automatic deletion of the archives.

Default Undefined (no limit to the age of `lsb.acct`).

LSB_ACCT_MAX_FILES

Syntax LSB_ACCT_MAX_FILES=*integer*

Description Enables automatic deletion of archived LSF accounting log files and specifies the archive limit.

Compatibility LSB_ACCT_SIZE or LSB_ACCT_AGE should also be defined.

Example `LSB_ACCT_MAX_FILES=10`

LSF maintains the current `lsb.acct` and up to 10 archives. Every time the old `lsb.acct.9` becomes `lsb.acct.10`, the old `lsb.acct.10` gets deleted.

Default Undefined (no deletion of `lsb.acct.`*n* files).

LSB_ACCT_SIZE

Syntax *kilobytes*

Description Enables automatic archiving of LSF accounting log files, and specifies the archive threshold. LSF archives the current log file if its size exceeds the specified number of kilobytes.

See Also LSB_ACCT_AGE also enables automatic archiving.

LSB_ACCT_MAX_FILES enables automatic deletion of the archives.

Default Undefined (no limit to the size of `lsb.acct`).

MAX_JOB_ARRAY_SIZE

Syntax **MAX_JOB_ARRAY_SIZE =** *integer*

Description Specifies the maximum index value of a job array that can be created by a user for a single job submission. The maximum number of jobs in a job array cannot exceed this value, and will be less if some index values are not used (start, end, and step values can all be used to limit the indices used in a job array).

A large job array allows a user to submit a large number of jobs to the system with a single job submission.

Specify an integer value from 1 to 65534.

Default 1000

MAX_JOB_ATTA_SIZE

Syntax **MAX_JOB_ATTA_SIZE =** *integer* | **0**

Specify any number less than 20000.

Description Maximum attached data size, in KB, that can be transferred to a job.

Maximum size for data attached to a job with the `bpost`(1) command. Useful if you use `bpost`(1) and `bread`(1) to transfer large data files between jobs and you want to limit the usage in the current working directory.

0 indicates that jobs cannot accept attached data files.

Default Undefined. LSF does not set a maximum size of job attachments.

MAX_JOBID

Syntax **MAX_JOBID=***integer*

Description The job ID limit. The job ID limit is the highest job ID that LSF will ever assign, and also the maximum number of jobs in the system.

Specify any integer from 999999 to 9999999 (for practical purposes, any seven-digit integer).

Example MAX_JOBID=1234567

Default 999999

MAX_JOB_MSG_NUM

Syntax **MAX_JOB_MSG_NUM =** *integer* | **0**

Description Maximum number of message slots for each job. Maximum number of messages that can be posted to a job with the `bpost`(1) command.

0 indicates that jobs cannot accept external messages.

Default 128

MAX_JOB_NUM

Syntax **MAX_JOB_NUM =** *integer*

Description The maximum number of finished jobs whose events are to be stored in the `lsb.events` log file.

Once the limit is reached, MBD starts a new event log file. The old event log file is saved as `lsb.events.`*n*, with subsequent sequence number suffixes incremented by 1 each time a new log file is started. Event logging continues in the new `lsb.events` file.

Default 1000

MAX_PREEXEC_RETRY

Syntax **MAX_PREEXEC_RETRY =** *integer*

Description MultiCluster only. The maximum number of times to attempt the pre-execution command of a job from a remote cluster.

If the job's pre-execution command fails all attempts, the job is returned to the submission cluster. The submission cluster will forward the job to one cluster at a time.

MAX_SBD_FAIL

Syntax **MAX_SBD_FAIL =** *integer*

Description The maximum number of retries for reaching a non-responding slave batch daemon, SBD.

The interval between retries is defined by MBD_SLEEP_TIME. If MBD fails to reach a host and has retried MAX_SBD_FAIL times, the host is considered unavailable. When a host becomes unavailable, MBD assumes that all jobs running on that host have exited and that all rerunnable jobs (jobs submitted with the `bsub -r` option) are scheduled to be rerun on another host.

Default 3

MAX_USER_PRIORITY

Syntax **MAX_USER_PRIORITY=***integer*

Description Enables user-assigned job priority and specifies the maximum job priority a user can assign to a job.

LSF administrators can assign a job priority higher than the specified value.

Compatibility User-assigned job priority changes the behavior of `btop` and `bbot`.

Example `MAX_USER_PRIORITY=100`

Specifies that 100 is the maximum job priority that can be specified by a user.

Default Undefined

See Also bsub, bmod, btop, bbot, "JOB_PRIORITY_OVER_TIME" on page 295

MBD_REFRESH_TIME

Syntax **`MBD_REFRESH_TIME =`** *seconds*

Description Time interval, in seconds, at which MBD will fork a new child MBD to service query requests to keep information sent back to clients updated. A child MBD processes query requests creating threads.

MBD_REFRESH_TIME applies only to UNIX platforms that support thread programming.

MBD_REFRESH_TIME works in conjunction with LSB_QUERY_PORT in `lsf.conf`. The child MBD continues to listen to the port number specified by LSB_QUERY_PORT and creates threads to service requests until the job changes status, a new job is submitted, or MBD_REFRESH_TIME has expired.

- If MBD_REFRESH_TIME is < 10 seconds, the child MBD exits at MBD_REFRESH_TIME even if the job changes status or a new job is submitted before MBD_REFRESH_TIME expires
- If MBD_REFRESH_TIME > 10 seconds, the child MBD exits at 10 seconds even if the job changes status or a new job is submitted before the 10 seconds
- If MBD_REFRESH_TIME > 10 seconds and no job changes status or a new job is submitted, the child MBD exits at MBD_REFRESH_TIME

The value of this parameter must be between 5 and 300. Any values specified out of this range are ignored, and the system default value is applied.

The `bjobs` command may not display up-to-date information if two consecutive query commands are issued before a child MBD expires because child MBD job information is not updated. If you use the `bjobs` command and do not get up-to-date information, you may need to decrease the value of this parameter. Note, however, that the lower the value of this parameter, the more you negatively affect performance.

The number of concurrent requests is limited by the number of concurrent threads that a process can have. This number varies by platform:

- Sun Solaris, 2500 threads per process
- AIX, 512 threads per process
- Digital, 256 threads per process
- HP-UX, 64 threads per process

Default 5 seconds if not defined or if defined value is less than 5; 300 seconds if defined value is more than 300

MBD_SLEEP_TIME

Syntax **`MBD_SLEEP_TIME =`** *seconds*

Description The job dispatching interval; how often LSF tries to dispatch pending jobs.

Default 60

MC_RUSAGE_UPDATE_INTERVAL

Syntax **`MC_RUSAGE_UPDATE_INTERVAL =`** *seconds*

Description MultiCluster only. If defined, enables resource use updating for MultiCluster jobs running in the cluster and specifies how often to update the information in the submission cluster.

You should use the same value for MBD_SLEEP_TIME and MC_RUSAGE_UPDATE_INTERVAL.

Default Undefined

NQS_QUEUES_FLAGS

Syntax **NQS_QUEUES_FLAGS =** *integer*

Description For Cray NQS compatibility only. Used by LSF to get the NQS queue information.

If the NQS version on a Cray is NQS 1.1, 80.42 or NQS 71.3, this parameter does not need to be defined.

For other versions of NQS on Cray, define both NQS_QUEUES_FLAGS and NQS_REQUESTS_FLAGS.

To determine the value of this parameter, run the NQS `qstat` command. The value of `Npk_int[1]` in the output is the value you need for this parameter. Refer to the NQS chapter in the *LSF Administrator's Guide* for more details.

Default Undefined

NQS_REQUESTS_FLAGS

Syntax **NQS_REQUESTS_FLAGS =** *integer*

Description For Cray NQS compatibility only.

If the NQS version on a Cray is NQS 80.42 or NQS 71.3, this parameter does not need to be defined.

If the version is NQS 1.1 on a Cray, set this parameter to 251918848. This is the is the `qstat` flag which LSF uses to retrieve requests on Cray in long format.

For other versions of NQS on a Cray, run the NQS `qstat` command. The value of `Npk_int[1]` in the output is the value you need for this parameter. Refer to the NQS chapter in the *LSF Administrator's Guide* for more details.

Default Undefined

PG_SUSP_IT

Syntax **PG_SUSP_IT =** *seconds*

Description The time interval that a host should be interactively idle (it > 0) before jobs suspended because of a threshold on the pg load index can be resumed.

This parameter is used to prevent the case in which a batch job is suspended and resumed too often as it raises the paging rate while running and lowers it while suspended. If you are not concerned with the interference with interactive jobs caused by paging, the value of this parameter may be set to 0.

Default 180 (seconds)

PREEMPTABLE_RESOURCES

Syntax **PREEMPTABLE_RESOURCES=***resource_name...*

Description Enables resource preemption when preemptive scheduling is enabled (has no effect if PREEMPTIVE is not also specified) and specifies the preemption resources. Specify shared numeric resources, static or decreasing, that LSF is configured to release (RELEASE=Y in `lsf.shared`, which is the default).

You must also configure LSF's preemption action to make the preempted application releases its resources. To kill preempted jobs instead of suspending them, set TERMINATE_WHEN=PREEMPT in `lsb.queues`, or set JOB_CONTROLS in `lsb.queues` and specify `brequeue` as the SUSPEND action.

Default Undefined (if preemptive scheduling is configured, LSF preempts on job slots only).

PREEMPT_FOR

Syntax **PREEMPT_FOR=[HOST_JLU | USER_JLP | GROUP_MAX |GROUP_JLP]...**

Description If preemptive scheduling is enabled, LSF does not count suspended jobs against the specified job slot limits. Specify a space-separated list of the following keywords:

- GROUP_MAX - total job slot limit for user groups, specified at the user level (MAX_JOBS in `lsb.users`); if preemptive scheduling is enabled, suspended jobs never count against the limit for individual users
- HOST_JLU - total number of jobs for users and user groups, specified at the host level (JL/U in `lsb.hosts`)
- USER_JLP - user-processor job slot limit for individual users, specified at the user level (JL/P in `lsb.users`)
- GROUP_JLP - per-processor job slot limit for user groups, specified at the user level (JL/P in `lsb.users`)

Job slot limits specified at the queue level are never affected by preemptive scheduling.

Default Undefined. If preemptive scheduling is configured, LSF preempts on the following limits only:

- total job slot limit for hosts, specified at the host level (MXJ in `lsb.hosts`)
- total job slot limit for individual users, specified at the user level (MAX_JOBS in `lsb.users`); by default, suspended jobs still count against the limit for user groups

PREEMPTION_WAIT_TIME

Syntax **PREEMPTION_WAIT_TIME =** *seconds*

Description Used only with resource preemption (you must specify PREEMPTABLE_RESOURCES in `lsb.params`).

The amount of time LSF waits, after preempting jobs, for preemption resources to become available. Specify at least 300 seconds.

If LSF does not get the resources after this time, LSF might preempt more jobs.

Default 300 (5 minutes)

RUN_JOB_FACTOR

Syntax **RUN_JOB_FACTOR =** *number*

Description Used only with fairshare scheduling. Job slots weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the number of job slots reserved and in use by a user.

Default 3.0

RUN_TIME_FACTOR

Syntax **RUN_TIME_FACTOR =** *number*

Description Used only with fairshare scheduling. Run time weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the total run time of a user's running jobs.

Default 0.7

SBD_SLEEP_TIME

Syntax **SBD_SLEEP_TIME =** *seconds*

Description The interval at which LSF checks the load conditions of each host, to decide whether jobs on the host must be suspended or resumed.

Default 30

SHARED_RESOURCE_UPDATE_FACTOR

Syntax **SHARED_RESOURCE_UPDATE_FACTOR =** *integer*

Description Determines the static shared resource update interval for the cluster.

Specify approximately how many times to update static shared resources during one MBD sleep time period. The formula is:

interval = MBD_SLEEP_TIME / SHARED_RESOURCE_UPDATE_FACTOR

where the result of the calculation is truncated to an integer. The static shared resource update interval is in seconds.

Default Undefined (all resources are updated only once, at the start of each dispatch turn).

SYSTEM_MAPPING_ACCOUNT

Syntax **SYSTEM_MAPPING_ACCOUNT =** *user_account*

Description (LSF Windows NT Workgroup installations only) User account to which all Windows NT workgroup user accounts are mapped.

Default Undefined

lsb.queues

Overview The `lsb.queues` file defines the batch queues in an LSF cluster.

This file is optional; if no queues are configured, LSF creates a queue named `default`, with all parameters set to default values.

Contents

lsb.queues Structure

Each queue definition begins with the line Begin Queue and ends with the line End Queue. The queue name must be specified; all other parameters are optional.

ADMINISTRATORS

Syntax **ADMINISTRATORS =** *user_name* | *user_group* ...

Description List of queue administrators.

Queue administrators can perform operations on any user's job in the queue, as well as on the queue itself.

Default Undefined (you must be a cluster administrator to operate on this queue).

BACKFILL

Syntax **BACKFILL = Y** | **N**

Description If Y, enables backfill scheduling for the queue.

A possible conflict exists if BACKFILL and PREEMPTION are specified together. A backfill queue cannot be preemptable. Therefore, if BACKFILL is enabled, do not also specify PREEMPTION = PREEMPTABLE.

Default Undefined (no backfilling).

CHKPNT

Syntax **CHKPNT =** *chkpnt_dir* [*chkpnt_period*]

Description Enables automatic checkpointing.

The checkpoint directory is the directory where the checkpoint files are created. Specify an absolute path or a path relative to CWD, do not use environment variables.

Specify the checkpoint period in minutes.

Job-level checkpoint parameters override queue-level checkpoint parameters.

Default Undefined.

CHUNK_JOB_SIZE

Syntax **CHUNK_JOB_SIZE =** *integer*

Description The maximum number of jobs allowed to be dispatched together in a chunk job. All of the jobs in the chunk are scheduled and dispatched as a unit, rather than individually.

Specify a positive integer greater than 1.

Use the CHUNK_JOB_SIZE parameter to configure queues that accept small, short-running jobs. The ideal candidates for job chunking are jobs that have the same host and resource requriements and typically take 1 to 2 minutes to run.

Job chunking can have the following advantages:

- Reduces communication between SBD and MBD and reduces scheduling overhead in MBD.
- Increases job throughput in MBD and CPU utilization on the execution hosts.

Affect on Job Throughput Note that throughput can deteriorate if the chunk job size is too big. Performance may decrease on queues with CHUNK_JOB_SIZE greater than 30. You should evaluate the chunk job size on your own systems for best performance.

Compatibility This parameter is ignored in the following kinds of queues:

- Interactive (INTERACTIVE = ONLY parameter)
- MultiCluster (SNDJOBS_TO or RCVJOBS_FROM parameters)

- Checkpointable (CHKPNT parameter)
- CPU limit greater than 30 minutes (CPULIMIT parameter)
- Run limit greater than 30 minutes (RUNLIMIT parameter)

Example The following configures a queue named `chunk`, which accepts up to 4 jobs in a chunk:

```
Begin Queue
QUEUE_NAME     = chunk
PRIORITY       = 50
CHUNK_JOB_SIZE = 4
End Queue
```

Default Undefined.

CORELIMIT

Syntax **CORELIMIT =** *integer*

Description The per-process (hard) core file size limit (in KB) for all of the processes belonging to a job from this queue (see `getrlimit(2)`).

Default Unlimited

CPULIMIT

Syntax **CPULIMIT =** [*default_limit*] *maximum_limit*

where *default_limit* and *maximum_limit* are:

[*hours*:]*minutes*[/*host_name* | /*host_model*]

Description Maximum normalized CPU time and optionally, the default normalized CPU time allowed for all processes of a job running in this queue. The name of a host or host model specifies the CPU time normalization host to use.

If a job dynamically spawns processes, the CPU time used by these processes is accumulated over the life of the job.

Processes that exist for fewer than 30 seconds may be ignored.

By default, if a default CPU limit is specified, jobs submitted to the queue without a job-level CPU limit are killed when the default CPU limit is reached.

If you specify only one limit, it is the maximum, or hard, CPU limit. If you specify two limits, the first one is the default, or soft, CPU limit, and the second one is the maximum CPU limit. The number of minutes may be greater than 59. Therefore, three and a half hours can be specified either as 3:30 or 210.

On Windows, a job which runs under a CPU time limit may exceed that limit by up to SBD_SLEEP_TIME. This is because SBD periodically checks if the limit has been exceeded. On UNIX systems, the CPU limit can be enforced by the operating system at the process level.

You can define whether the CPU limit is a per-process limit enforced by the OS or a per-job limit enforced by LSF with LSB_JOB_CPULIMIT in `lsf.conf`.

Default Unlimited

DATALIMIT

Syntax **`DATALIMIT =`** [*default_limit*] *maximum_limit*

Description The per-process data segment size limit (in KB) for all of the processes belonging to a job from this queue (see `getrlimit(2)`).

By default, if a default data limit is specified, jobs submitted to the queue without a job-level data limit are killed when the default data limit is reached.

If you specify only one limit, it is the maximum, or hard, data limit. If you specify two limits, the first one is the default, or soft, data limit, and the second one is the maximum data limit

Default Unlimited

DEFAULT_HOST_SPEC

Syntax **DEFAULT_HOST_SPEC =** *host_name* | *host_model*

Description The default CPU time normalization host for the queue.

The CPU factor of the specified host or host model will be used to normalize the CPU time limit of all jobs in the queue, unless the CPU time normalization host is specified at the job level.

Default Undefined.

DESCRIPTION

Syntax **DESCRIPTION =** *text*

Description Description of the job queue that will be displayed by `bqueues -l`.

This description should clearly describe the service features of this queue, to help users select the proper queue for each job.

The text can include any characters, including white space. The text can be extended to multiple lines by ending the preceding line with a backslash (\). The maximum length for the text is 512 characters.

DISPATCH_WINDOW

Syntax **DISPATCH_WINDOW =** *time_window* ...

Description The time windows in which jobs from this queue are dispatched. Once dispatched, jobs are no longer affected by the dispatch window.

Default Undefined (always open).

EXCL_RMTJOB

Syntax **EXCL_RMTJOB = Y** | **N**

Description MultiCluster only. If Y, enables preferential local scheduling.

It prevents the queue from accepting MultiCluster jobs unless it can run them on the next dispatch turn. It rejects MultiCluster jobs if either of the following conditions is true:

- The execution queue has one or more local jobs pending.
- The execution queue has no job slots available and the MultiCluster job cannot preempt a running job.

For every remote job that is rejected, it causes that remote cluster to temporarily stop submitting jobs to the local cluster.

Default Undefined (no preferential local scheduling).

EXCLUSIVE

Syntax **EXCLUSIVE = Y | N**

Description If Y, specifies an exclusive queue.

Jobs submitted to an exclusive queue with `bsub -x` will only be dispatched to a host that has no other LSF jobs running.

FAIRSHARE

Description Enables queue-level fairshare and specifies share assignments. Only users with share assignments can submit jobs to the queue.

Syntax **FAIRSHARE = USER_SHARES[[***user***, *number_shares***]** ...**]**

- Specify at least one user share assignment.
- Enclose the list in square brackets, as shown.
- Enclose each user share assignment in square brackets, as shown.
- *user*

  Specify users who are also configured to use queue. You can assign the shares to:

  - A single user (specify *user_name*)
  - Users in a group, individually (specify *group_name*@) or collectively (specify *group_name*)

- Users not included in any other share assignment, individually (specify the keyword `default`) or collectively (specify the keyword `others`)

By default, when resources are assigned collectively to a group, the group members compete for the resources on a first-come, first-served (FCFS) basis. You can use hierarchical fairshare to further divide the shares among the group members.

When resources are assigned to members of a group individually, the share assignment is recursive. Members of the group and of all subgroups always compete for the resources according to FCFS scheduling, regardless of hierarchical fairshare policies.

- *number_shares*

Specify a positive integer representing the number of shares of the cluster resources assigned to the user.

The number of shares assigned to each user is only meaningful when you compare it to the shares assigned to other users or to the total number of shares. The total number of shares is just the sum of all the shares assigned in each share assignment.

Compatibility Do not configure hosts in a cluster to use fairshare at both queue and host levels.

Default Undefined (no fairshare).

FILELIMIT

Syntax **`FILELIMIT =`** *integer*

Description The per-process (hard) file size limit (in KB) for all of the processes belonging to a job from this queue (see `getrlimit(2)`).

Default Unlimited

HJOB_LIMIT

Syntax **`HJOB_LIMIT =`** *integer*

Description Per-host job slot limit.

Maximum number of job slots that this queue can use on any host. This limit is configured per host, regardless of the number of processors it may have.

This may be useful if the queue dispatches jobs that require a node-locked license. If there is only one node-locked license per host then the system should not dispatch more than one job to the host even if it is a multiprocessor host.

Example The following will run a maximum of one job on each of hostA, hostB, and `hostC`:

```
Begin Queue
...
HJOB_LIMIT = 1
HOSTS=hostA hostB hostC
...
End Queue
```

Default Unlimited

HOSTS

Syntax **`HOSTS =`** [**~**]*host_name*[**+***pref_level*] | *host_partition*[**+***pref_level*] | [**~**]*host_group*[**+***pref_level*] | **`others`**[**+***pref_level*] | **`all`** | **`none`** ...

Description A space-separated list of hosts, host groups, and host partitions on which jobs from this queue can be run. All the members of the host list should either belong to a single host partition or not belong to any host partition. Otherwise, job scheduling may be affected.

Any item can be followed by a plus sign (+) and a positive number to indicate the preference for dispatching a job to that host, host group, or host partition. A higher number indicates a higher preference. If a host preference is not given, it is assumed to be 0. Hosts at the same level of preference are ordered by load.

Use the keyword `others` to indicate all hosts not explicitly listed.

Use the not operator (~) to exclude hosts or host groups from the queue. This is useful if you have a large cluster but only want to exclude a few hosts from the queue definition.

Use the keyword `all` to indicate all hosts not explicitly excluded.

If you have LSF MultiCluster and want to make a remote execution queue, use the keyword `none` to indicate that no local host will be used to run jobs in this queue, and use SNDJOBS_TO to specify at least one remote execution queue.

Compatibility Host preferences specified by `bsub -m` override the queue specification.

Example 1

```
HOSTS = hostA+1 hostB hostC+1 GroupX+3
```

This example defines three levels of preferences: run jobs on hosts in GroupX as much as possible, otherwise run on either `hostA` or `hostC` if possible, otherwise run on `hostB`. Jobs should not run on `hostB` unless all other hosts are too busy to accept more jobs.

Example 2

```
HOSTS = hostD+1 others
```

Run jobs on `hostD` as much as possible, otherwise run jobs on the least-loaded host available.

Example 3

```
HOSTS = Group1 ~hostA hostB hostC
```

Run jobs on `hostB`, `hostC`, and all hosts in `Group1` except for `hostA`.

Example 4

```
HOSTS = all ~group2 ~hostA
```

Run jobs on all hosts in the cluster, except for `hostA` and the hosts in `group2`.

Default `all` (the queue can use all hosts in the cluster, and every host has equal preference).

IGNORE_DEADLINE

Syntax **`IGNORE_DEADLINE = Y`**

Description If Y, disables deadline constraint scheduling (starts all jobs regardless of deadline contraints).

IMPT_JOBBKLG

Syntax **IMPT_JOBBKLG =** *integer*

Description MultiCluster only. Enables MultiCluster job threshold.

It limits the number of jobs from remote clusters that can be pending in the queue. For every remote job that is rejected, it causes that remote cluster to temporarily stop submitting jobs to the local cluster.

A value of 0 or less means the queue will never accept a job from a remote cluster.

Default Undefined (accepts an unlimited number of MultiCluster jobs).

INTERACTIVE

Syntax **INTERACTIVE = NO** | **ONLY**

Description Causes the queue to reject interactive batch jobs (NO) or accept nothing but interactive batch jobs (ONLY).

Interactive batch jobs are submitted via `bsub -I`.

Default Undefined (the queue accepts both interactive and non-interactive jobs).

JOB_ACCEPT_INTERVAL

Syntax **JOB_ACCEPT_INTERVAL =** *integer*

Description The number of dispatch turns to wait after dispatching a job to a host, before dispatching a second job to the same host. By default, a dispatch turn lasts 60 seconds (MBD_SLEEP_TIME in `lsb.params`).

If 0 (zero), a host may accept more than one job in each dispatch turn. By default, there is no limit to the total number of jobs that can run on a host, so if this parameter is set to 0, a very large number of jobs might be dispatched to a host all at once. You may notice performance problems if this occurs.

JOB_ACCEPT_INTERVAL set at the queue level (`lsb.queues`) overrides JOB_ACCEPT_INTERVAL set at the cluster level (`lsb.params`).

Default Undefined (the queue uses JOB_ACCEPT_INTERVAL defined in `lsb.params`, which has a default value of 1).

JOB_CONTROLS

Syntax **JOB_CONTROLS = SUSPEND**[*signal* | *command* | **CHKPNT**] **RESUME**[*signal* | *command*] **TERMINATE**[*signal* | *command* | **CHKPNT**]

- CHKPNT is a special action, which causes the system to checkpoint the job. If the SUSPEND action is CHKPNT, the job is checkpointed and then stopped by sending the SIGSTOP signal to the job automatically.
- *signal* is a UNIX signal name (such as SIGSTOP or SIGTSTP).
- *command* specifies a `/bin/sh` command line to be invoked. Do not specify a signal followed by an action that triggers the same signal (for example, do not specify `JOB_CONTROLS=TERMINATE[bkill]` or `JOB_CONTROLS=TERMINATE[brequeue]`). This will cause a deadlock between the signal and the action.

Description Changes the behaviour of the SUSPEND, RESUME, and TERMINATE actions in LSF.

For SUSPEND and RESUME, if the action is a command, the following points should be considered:

- The contents of the configuration line for the action are run with `/bin/sh -c` so you can use shell features in the command.
- The standard input, output, and error of the command are redirected to the NULL device.
- The command is run as the user of the job.

- All environment variables set for the job are also set for the command action. The following additional environment variables are set:
  - LSB_JOBPGIDS — a list of current process group IDs of the job
  - LSB_JOBPIDS —a list of current process IDs of the job

  For the SUSPEND action command, the following environment variable is also set:
  - LSB_SUSP_REASONS — an integer representing a bitmap of suspending reasons as defined in `lsbatch.h`

    The suspending reason can allow the command to take different actions based on the reason for suspending the job.

Default On UNIX, by default, SUSPEND sends SIGTSTP for parallel or interactive jobs and SIGSTOP for other jobs. RESUME sends SIGCONT. TERMINATE sends SIGINT, SIGTERM and SIGKILL in that order.

On Windows NT, actions equivalent to the UNIX signals have been implemented to do the default job control actions. Job control messages replace the SIGINT and SIGTERM signals, but only customized applications will be able to process them. Termination is implemented by the `TerminateProcess( )` system call.

JOB_STARTER

Syntax **`JOB_STARTER =`** *starter* [*starter*] [**`"%USRCMD"`**] [*starter*]

Description Creates a specific environment for submitted jobs prior to execution.

starter is any executable that can be used to start the job (i.e., can accept the job as an input argument). Optionally, additional strings can be specified.

By default, the user commands run after the job starter. A special string, %USRCMD, can be used to represent the position of the user's job in the job starter command line. The %USRCMD string may be enclosed with quotes or followed by additional commands.

Example

```
JOB_STARTER = csh -c "%USRCMD;sleep 10"
```

In this case, if a user submits a job

```
% bsub myjob arguments
```

the command that actually runs is:

```
% csh -c "myjob arguments;sleep 10"
```

Default Undefined (no job starter).

load_index

Syntax *load_index* **=** *loadSched*[**/***loadStop*]

Specify `io`, `it`, `ls`, `mem`, `pg`, `r15s`, `r1m`, `r15m`, `swp`, `tmp`, `ut`, or a non-shared custom external load index. Specify multiple lines to configure thresholds for multiple load indices.

Specify `io`, `it`, `ls`, `mem`, `pg`, `r15s`, `r1m`, `r15m`, `swp`, `tmp`, `ut`, or a non-shared custom external load index as a column. Specify multiple columns to configure thresholds for multiple load indices.

Description Scheduling and suspending thresholds for the specified dynamic load index.

The `loadSched` condition must be satisfied before a job is dispatched to the host. If a RESUME_COND is not specified, the `loadSched` condition must also be satisfied before a suspended job can be resumed.

If the `loadStop` condition is satisfied, a job on the host will be suspended.

The `loadSched` and `loadStop` thresholds permit the specification of conditions using simple AND/OR logic. Any load index that does not have a configured threshold has no effect on job scheduling.

LSF will not suspend a job if the job is the only batch job running on the host and the machine is interactively idle (`it`>0).

The `r15s`, `r1m`, and `r15m` CPU run queue length conditions are compared to the effective queue length as reported by `lsload -E`, which is normalized for multiprocessor hosts. Thresholds for these parameters should be set at appropriate levels for single processor hosts.

Example
```
MEM=100/10
SWAP=200/30
```

These two lines translate into a `loadSched` condition of

```
mem>=100 && swap>=200
```

and a `loadStop` condition of

```
mem < 10 || swap < 30
```

Default Undefined.

MAX_RSCHED_TIME

Syntax **MAX_RSCHED_TIME =** *value* | **infinit**

Description MultiCluster only. Number of job dispatching intervals in a MultiCluster scheduling cycle. The length of a MultiCluster job scheduling cycle in seconds is:

```
MAX_RSCHED_TIME * MBD_SLEEP_TIME = job_scheduling_cycle
```

Specify `infinit` to disable remote timeout and to make sure that jobs always get dispatched in the correct order. When you specify `infinit`, MBD waits indefinitely for allocation reply from the execution MBD. This keeps remote jobs from being executed in FCFS order. If the execution MBD is restarted, timeout will be set, and these jobs will be rescheduled by the local MBD.

Default 20 (20 minutes by default).

MC_FAST_SCHEDULE

Syntax **MC_FAST_SCHEDULE =** [**y** | **n**]

Description MultiCluster only.

Specify `y` to enable preferential MultiCluster scheduling.

By default, jobs forwarded from a remote cluster are treated just like jobs submitted to the local queue, and wait in the pending job list for the next dispatch turn.

If resource requirements are not important, you can give preference to remote jobs. If you enable preferential MultiCluster scheduling, jobs from a remote queue are dispatched to the execution host immediately, without evaluating the resource requirement, and without waiting for the next dispatch turn.

This helps to speed up MultiCluster operation, but jobs might fail because their resource requirement is ignored.

Default Undefined (preferential MultiCluster scheduling is disabled).

MEMLIMIT

Syntax **MEMLIMIT =** [*default_limit*] *maximum_limit*

Description The per-process (hard) process resident set size limit (in KB) for all of the processes belonging to a job from this queue (see `getrlimit(2)`).

Sets the maximum amount of physical memory (resident set size, RSS) that may be allocated to a process.

By default, if a default memory limit is specified, jobs submitted to the queue without a job-level memory limit are killed when the default memory limit is reached.

If you specify only one limit, it is the maximum, or hard, memory limit. If you specify two limits, the first one is the default, or soft, memory limit, and the second one is the maximum memory limit.

LSF has two methods of enforcing memory usage:

- OS Memory Limit Enforcement
- LSF Memory Limit Enforcement

OS Memory Limit Enforcement OS memory limit enforcement is the default MEMLIMIT behavior and does not require further configuration. OS enforcement usually allows the process to eventually run to completion. LSF passes MEMLIMIT to the OS which uses it as a guide for the system scheduler and memory allocator. The system may allocate more memory to a process if there is a surplus. When memory is low, the system takes memory from and

lowers the scheduling priority (re-nice) of a process that has exceeded its declared MEMLIMIT. Only available on systems that support `RUSAGE_RSS` for `setrlimit()`. Not supported on:

- Sun Solaris 2.x
- Windows NT

LSF Memory Limit Enforcement

To enable LSF memory limit enforcement, set LSB_MEMLIMIT_ENFORCE in `lsf.conf` to `y`. LSF memory limit enforcement explicitly sends a signal to kill a running process once it has allocated memory past MEMLIMIT.

You can also enable LSF memory limit enforcement by setting LSB_JOB_MEMLIMIT in `lsf.conf` to `y`. The difference between LSB_JOB_MEMLIMIT set to y and LSB_MEMLIMIT_ENFORCE set to y is that with LSB_JOB_MEMLIMIT, only the per-job memory limit enforced by LSF is enabled. The per-process memory limit enforced by the OS is disabled. With LSB_MEMLIMIT_ENFORCE set to y, both the per-job memory limit enforced by LSF and the per-process memory limit enforced by the OS are enabled.

Available for all systems on which LSF collects total memory usage.

Example

The following configuration defines a queue with a memory limit of 5000 KB:

```
Begin Queue
QUEUE_NAME  = default
DESCRIPTION = Queue with memory limit of 5000 kbytes
MEMLIMIT    = 5000
End Queue
```

Default

Unlimited

MIG

Syntax

`MIG =` *minutes*

Description

Enables automatic job migration and specifies the migration threshold, in minutes.

If a checkpointable or rerunnable job dispatched to the host is suspended (SSUSP state) for longer than the specified number of minutes, the job is migrated (unless another job on the same host is being migrated). A value of 0 specifies that a suspended job should be migrated immediately.

If a migration threshold is defined at both host and queue levels, the lower threshold is used.

Default Undefined (no automatic job migration).

NEW_JOB_SCHED_DELAY

Syntax **NEW_JOB_SCHED_DELAY =** *seconds*

Description The maximum or minimum length of time that a new job waits before being dispatched; the behavior depends on whether the delay period specified is longer or shorter than a regular dispatch interval (MBD_SLEEP_TIME in `lsb.params`, 60 seconds by default).

- If less than the dispatch interval, specifies the maximum number of seconds to wait, after a new job is submitted, before starting a new dispatch turn and scheduling the job. Usually, this causes LSF to schedule dispatch turns more frequently. You might notice performance problems (affecting the entire cluster) if this value is set too low in a busy queue.
- If 0, starts a new dispatch turn as soon as a job is submitted to this queue (affecting the entire cluster).
- If greater than the dispatch interval, specifies the minimum number of seconds to wait, after a new job is submitted, before scheduling the job. Has no effect of the timing of the dispatch turns, but new jobs in this queue are always delayed by one or more dispatch turns.

Default 10 seconds.

NICE

Syntax **NICE =** *integer*

Description Adjusts the UNIX scheduling priority at which jobs from this queue execute.

The default value of 0 maintains the default scheduling priority for UNIX interactive jobs. This value adjusts the run-time priorities for batch jobs on a queue-by-queue basis, to control their effect on other batch or interactive jobs. See the `nice(1)` manual page for more details.

On Windows NT, this value is mapped to Windows NT process priority classes as follows:

- `nice>=0` corresponds to an NT priority class of `IDLE`
- `nice<0` corresponds to an NT priority class of `NORMAL`

LSF on Windows NT does not support `HIGH` or `REAL-TIME` priority classes.

Default 0

NQS_QUEUES

Syntax **`NQS_QUEUES =`** *NQS_ queue_name@NQS_host_name ...*

Description Makes the queue an NQS forward queue.

NQS_host_name is an NQS host name that can be the official host name or an alias name known to the LSF master host through `gethostbyname(3)`.

NQS_queue_name is the name of an NQS destination queue on this host. NQS destination queues are considered for job routing in the order in which they are listed here. If a queue accepts the job, it is routed to that queue. If no queue accepts the job, it remains pending in the NQS forward queue.

`lsb.nqsmaps` must be present for LSF Batch to route jobs in this queue to NQS systems.

Since many features of LSF are not supported by NQS, the following queue configuration parameters are ignored for NQS forward queues: PJOB_LIMIT, POLICIES, RUN_WINDOW, DISPATCH_WINDOW,

RUNLIMIT, HOSTS, MIG. In addition, scheduling load threshold parameters are ignored because NQS does not provide load information about hosts.

Default Undefined.

PJOB_LIMIT

Syntax **PJOB_LIMIT =** *integer*

Description Per-processor job slot limit for the queue.

Maximum number of job slots that this queue can use on any processor. This limit is configured per processor, so that multiprocessor hosts automatically run more jobs.

Default Unlimited

POST_EXEC

Syntax **POST_EXEC =** *command*

Description A command run on the execution host after the job.

UNIX The entire contents of the configuration line of the pre- and post-execution commands are run under `/bin/sh -c`, so shell features can be used in the command.

The pre- and post-execution commands are run in `/tmp`.

Standard input and standard output and error are set to:

`/dev/null`

The output from the pre- and post-execution commands can be explicitly redirected to a file for debugging purposes.

The PATH environment variable is set to:

`'/bin /usr/bin /sbin/usr/sbin'`

Windows NT The pre- and post-execution commands are run under `cmd.exe/c`.

To run these commands under a different user account (such as root, to do privileged operations, if necessary), configure the parameter LSB_PRE_POST_EXEC_USER in `lsf.sudoers`.

Standard input and standard output and error are set to NUL. The output from the pre- and post-execution commands can be explicitly redirected to a file for debugging purposes.

The PATH is determined by the setup of the LSF Service.

- Other environment variables set for the job are also set for the pre- and post-execution commands.
- When a job is dispatched from a queue that has a pre-execution command, LSF Batch will remember the post-execution command defined for the queue from which the job is dispatched. If the job is later switched to another queue or the post-execution command of the queue is changed, LSF Batch will still run the original post-execution command for this job.
- When the post-execution command is run, the environment variable LSB_JOBEXIT_STAT is set to the exit status of the job. Refer to the manual page for `wait`(2) for the format of this exit status.
- The post-execution command is also run if a job is requeued because the job's execution environment fails to be set up or if the job exits with one of the queue's REQUEUE_EXIT_VALUES. The environment variable LSB_JOBPEND is set if the job is requeued. If the job's execution environment could not be set up, LSB_JOBEXIT_STAT is set to 0.

Default No post-execution commands

PRE_EXEC

Syntax **PRE_EXEC =** *command*

Description A command run on the execution host before the job.

To specify a pre-execution command at the job level, use `bsub -E`. If both queue and job level pre-execution commands are specified, the job level pre-execution is run after the queue level pre-execution command.

For UNIX:

- The entire contents of the configuration line of the pre- and post-execution commands are run under `/bin/sh -c`, so shell features can be used in the command.
- The pre- and post-execution commands are run in `/tmp`.
- Standard input and standard output and error are set to: `/dev/null`
- The output from the pre- and post-execution commands can be explicitly redirected to a file for debugging purposes.
- The PATH environment variable is set to:
  /bin /usr/bin /sbin/usr/sbin

For Windows NT:

- The pre- and post-execution commands are run under `cmd.exe/c`.
- To run these commands under a different user account (such as root, to do privileged operations, if necessary), configure the parameter LSB_PRE_POST_EXEC_USER in `lsf.sudoers`.
- Standard input and standard output and error are set to NUL. The output from the pre- and post-execution commands can be explicitly redirected to a file for debugging purposes.
- The PATH is determined by the setup of the LSF Service.

For UNIX and Windows NT:

- If the pre-execution command exits with a non-zero exit code, it is considered to have failed, and the job is requeued to the head of the queue. This feature can be used to implement customized scheduling by having the pre-execution command fail if conditions for dispatching the job are not met.
- Other environment variables set for the job are also set for the pre- and post-execution commands.

Default No pre-execution commands

PREEMPTION

Syntax **PREEMPTION = PREEMPTIVE[[***queue_name...***]] PREEMPTABLE[[***queue_name...***]]**

You can specify PREEMPTIVE or PREEMPTABLE or both.When you specify a list of queues, you must enclose the list in one set of square brackets.

Description Enables preemptive scheduling and defines a preemption policy for the queue.

- PREEMPTIVE defines a preemptive queue. Jobs in this queue preempt jobs from the specified lower-priority queues or from all lower-priority queues by default. If you specify a list of lower-priority queues, you must enclose the list in one set of square brackets.
- PREEMPTABLE defines a preemptable queue. Jobs in this queue can be preempted by jobs from specified higher-priority queues, or from all higher-priority queues by default, even if the higher-priority queues are not preemptive. If you specify a list of higher-priority queues, you must enclose the list in one set of square brackets.

PREEMPTIVE and PREEMPTABLE can be used together, to specify that jobs in this queue can always preempt jobs in lower priority queues and can always be preempted by jobs from higher priority queues.

PRIORITY

Syntax **PRIORITY =** *integer*

Description The queue priority. A higher value indicates a higher LSF dispatching priority, relative to other queues.

LSF schedules jobs from one queue at a time, starting with the the highest-priority queue. If multiple queues have the same priority, LSF schedules all the jobs from these queues in first-come, first-served order.

Queue priority in LSF is completely independent of the UNIX scheduler's priority system for time-sharing processes. In LSF, the NICE parameter is used to set the UNIX time-sharing priority for batch jobs.

Default 1 (lowest possible priority)

PROCESSLIMIT

Syntax **PROCESSLIMIT =** [*default_limit*] *maximum_limit*

Description Limits the number of concurrent processes that can be part of a job.

By default, if a default process limit is specified, jobs submitted to the queue without a job-level process limit are killed when the default process limit is reached.

If you specify only one limit, it is the maximum, or hard, process limit. If you specify two limits, the first one is the default, or soft, process limit, and the second one is the maximum process limit.

Default Unlimited

PROCLIMIT

Syntax **PROCLIMIT =** [*minimum_limit* [*default_limit*]] *maximum_limit*

Description Maximum number of slots that can be allocated to a job. For parallel jobs, the maximum number of processors that can be allocated to t he job.

Optionally specifies the minimum and default number of job slots.

Jobs that specify fewer slots than the minimum PROCLIMIT or more slots than the maximum PROCLIMIT cannot use this queue and are rejected.

All limits must be positive numbers greater than or equal to 1 that satisfy the following relationship:

1 <= *minimum* <= *default* <= *maximum*

You can specify up to three limits in the PROCLIMIT parameter:

If You Specify ...	Then ...
One limit	It is the maximum processor limit. The minimum and default limits are set to 1.
Two limits	The first is the minimum processor limit, and the second one is the maximum. The default is set equal to the minimum. The minimum must be less than or equal to the maximum.
Three limits	The first is the minimum processor limit, the second is the default processor limit, and the third is the maximum. The minimum must be less than the default and the maximum.

Default Unlimited, the default number of slots is 1.

QJOB_LIMIT

Syntax **QJOB_LIMIT =** *integer*

Description Job slot limit for the queue. Total number of job slots that this queue can use.

Default Unlimited

QUEUE_NAME

Syntax **QUEUE_NAME =** *string*

Description Required. Name of the queue.

Specify any ASCII string up to 40 characters long. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces. You cannot specify the reserved name `default`.

Default You must specify this parameter to define a queue. The default queue automatically created by LSF is named `default`.

RCVJOBS_FROM

Syntax **RCVJOBS_FROM =** *cluster_name ...* | **allclusters**

Description MultiCluster only. Defines a MultiCluster execution queue.

Specify cluster names, separated by a space. The administrator of each remote cluster determines which queues in that cluster will forward jobs to the local cluster.

Use the keyword `allclusters` to specify any remote cluster. If the remote cluster sends jobs to this queue, `bclusters` displays the queue status as `ok` or `disc`.

By default, queues will not run jobs in other clusters, even if MultiCluster has been installed. You must configure queues in both the submission and execution cluster to use LSF MultiCluster:

The submission queue must be configured to send jobs to the execution queue. The execution queue must be configured to accept jobs from the cluster that contains the submission queue.

You can configure multiple submission and execution queue pairs. A queue can submit to as many other queues as you want, and can accept jobs from as many other clusters as you want.

Example
```
RCVJOBS_FROM=cluster2 cluster4 cluster6
```

This queue accepts remote jobs from clusters 2, 4, and 6.

REQUEUE_EXIT_VALUES

Syntax **REQUEUE_EXIT_VALUES =** [*exit_code ...*] [**EXCLUDE(***exit_code ...***)**]

Description Enables automatic job requeue and sets the LSB_EXIT_REQUEUE environment variable.

Separate multiple exit codes with spaces. Define an exit code as EXCLUDE(*exit_code*) to enable exclusive job requeue. Exclusive job requeue does not work for MultiCluster jobs or parallel jobs.

Jobs are requeued to the head of the queue from which they were dispatched. The output from the failed run is not saved, and the user is not notified by LSF.

A job terminated by a signal is not requeued.

If MBD is restarted, it will not remember the previous hosts from which the job exited with an exclusive requeue exit code. In this situation, it is possible for a job to be dispatched to hosts on which the job has previously exited with an exclusive exit code.

In a MultiCluster environment, this parameter has no effect on jobs from a remote cluster, because LSF always forwards these jobs and their exit codes to the submission cluster.

Automatic job requeue and exclusive job requeue are described in the *LSF Administrator's Guide*.

Example `REQUEUE_EXIT_VALUES=30 EXCLUDE(20)`

means that jobs with exit code 30 are requeued, jobs with exit code 20 are requeued exclusively, and jobs with any other exit code are not requeued.

Default Undefined (jobs in this queue are not requeued)

RERUNNABLE

Syntax **`RERUNNABLE = yes | no`**

Description If `yes`, enables automatic job rerun (restart).

Default `no`

RES_REQ

Syntax **RES_REQ =** *res_req*

Description Resource requirements used to determine eligible hosts. Specify a resource requirement string as usual. The resource requirement string lets you specify conditions in a more flexible manner than using the load thresholds.

The `select` section defined at the queue level must be satisfied at in addition to any job-level requirements or load thresholds.

The `rusage` section defined at the queue level overrides the `rusage` section defined at the job level, and jobs are rejected if they specify resource reservation requirements that exceed the requirements specified at the queue level.

The `order` section defined at the queue level is ignored if any resource requirements are specified at the job level (if the job-level resource requirements do not include the `order` section, the default order, `r15s:pg`, is used instead of the queue-level resource requirement).

The `span` section defined at the queue level is ignored if the `span` section is also defined at the job level.

If RES_REQ is defined at the queue level and there are no load thresholds defined, the pending reasons for each individual load index will not be displayed by `bjobs`.

Default `select[type==local] order[r15s:pg]`. If this parameter is defined and a host model or Boolean resource is specified, the default type will be `any`.

RESUME_COND

Syntax **RESUME_COND =** *res_req*

Use the `select` section of the resource requirement string to specify load thresholds. All other sections are ignored.

Description LSF automatically resumes a suspended (SSUSP) job in this queue if the load on the host satisfies the specified conditions.

If RESUME_COND is not defined, then the `loadSched` thresholds are used to control resuming of jobs. The `loadSched` thresholds are ignored, when resuming jobs, if RESUME_COND is defined.

RUN_WINDOW

Syntax **RUN_WINDOW =** *time_window ...*

Description Time periods during which jobs in the queue are allowed to run.

When the window closes, LSF suspends jobs running in the queue and stops dispatching jobs from the queue. When the window reopens, LSF resumes the suspended jobs and begins dispatching additional jobs.

Default Undefined (queue is always active)

RUNLIMIT

Syntax **RUNLIMIT =** [*default_limit*] *maximum_limit*

where *default_limit* and *maximum_limit* are:

[*hours*:]*minutes*[*/host_name* | */host_model*]

Description The maximum run limit and optionally the default run limit. The name of a host or host model specifies the run time normalization host to use.

By default, jobs that are in the RUN state for longer than the specified maximum run limit are killed by LSF. You can optionally provide your own termination job action to override this default.

Jobs submitted with a job-level run limit (`bsub -W`) that is less than the maximum run limit are killed when their job-level run limit is reached. Jobs submitted with a run limit greater than the maximum run limit are rejected by the queue.

If a default run limit is specified, jobs submitted to the queue without a job-level run limit are killed when the default run limit is reached. The default run limit is used with backfill scheduling of parallel jobs.

If you specify only one limit, it is the maximum, or hard, run limit. If you specify two limits, the first one is the default, or soft, run limit, and the second one is the maximum run limit. The number of minutes may be greater than 59. Therefore, three and a half hours can be specified either as 3:30, or 210.

Default Unlimited

SLOT_RESERVE

Syntax **SLOT_RESERVE = MAX_RESERVE_TIME[***integer***]**

Description Enables processor reservation and specifies the number of dispatch turns over which a parallel job can reserve job slots.

After this time, if a job has not accumulated enough job slots to start, it releases all its reserved job slots. This means a job cannot reserve job slots for more than (*integer* * MBD_SLEEP_TIME) seconds.

MBD_SLEEP_TIME is defined in `lsb.params`; the default value is 60 seconds.

Example
```
SLOT_RESERVE = MAX_RESERVE_TIME[5]
```

This example specifies that parallel jobs have up to 5 dispatch turns to reserve sufficient job slots (equal to 5 minutes, by default).

Default Undefined (no processor reservation)

SNDJOBS_TO

Syntax **SNDJOBS_TO =** *queue_name*@*cluster_name* ...

Description Defines a MultiCluster submission queue.

Specify remote queue names, in the form *queue_name*@*cluster_name*, separated by a space.

Example
```
SNDJOBS_TO=queue2@cluster2 queue3@cluster2 queue3@cluster3
```

STACKLIMIT

Syntax **STACKLIMIT =** *integer*

Description The per-process (hard) stack segment size limit (in KB) for all of the processes belonging to a job from this queue (see `getrlimit(2)`).

Default Unlimited

STOP_COND

Syntax **STOP_COND =** *res_req*

Use the `select` section of the resource requirement string to specify load thresholds. All other sections are ignored.

Description LSF automatically suspends a running job in this queue if the load on the host satisfies the specified conditions.

- LSF will not suspend the only job running on the host if the machine is interactively idle (`it` > 0).
- LSF will not suspend a forced job (`brun -f`).
- LSF will not suspend a job because of paging rate if the machine is interactively idle.

If STOP_COND is specified in the queue and there are no load thresholds, the suspending reasons for each individual load index will not be displayed by `bjobs`.

Example

```
STOP_COND= select[((!cs && it < 5) || (cs && mem < 15 && swap < 50))]
```

In this example, assume “cs” is a Boolean resource indicating that the host is a computer server. The stop condition for jobs running on computer servers is based on the availability of swap memory. The stop condition for jobs running on other kinds of hosts is based on the idle time.

SWAPLIMIT

Syntax **SWAPLIMIT =** *integer*

Description The amount of total virtual memory limit (in KB) for a job from this queue.

This limit applies to the whole job, no matter how many processes the job may contain.

The action taken when a job exceeds its SWAPLIMIT or PROCESSLIMIT is to send SIGQUIT, SIGINT, SIGTERM, and SIGKILL in sequence. For CPULIMIT, SIGXCPU is sent before SIGINT, SIGTERM, and SIGKILL.

Default Unlimited

TERMINATE_WHEN

Description Configures the queue to invoke the TERMINATE action instead of the SUSPEND action in the specified circumstance.

Syntax **TERMINATE_WHEN = WINDOW | LOAD | PREEMPT**

- WINDOW — kills jobs if the run window closes.
- LOAD — kills jobs when the load exceeds the suspending thresholds.
- PREEMPT — kills jobs that are being preempted.

Example Set TERMINATE_WHEN to WINDOW to define a night queue that will kill jobs if the run window closes:

```
Begin Queue
NAME           = night
RUN_WINDOW     = 20:00-08:00
TERMINATE_WHEN = WINDOW
JOB_CONTROLS   = TERMINATE[kill -KILL $LS_JOBPGIDS; mail - s
"job $LSB_JOBID killed by queue run window" $USER < /dev/null]
End Queue
```

UJOB_LIMIT

Syntax **UJOB_LIMIT =** *integer*

Description Per-user job slot limit for the queue. Maximum number of job slots that each user can use in this queue.

Default Unlimited

USERS

Syntax **USERS = all** | *user_name* | *user_group* ...

Description A list of users or user groups that can submit jobs to this queue (if FAIRSHARE is also defined, only users defined by both parameters can submit jobs).

Use the reserved word `all` to specify all LSF users.

LSF cluster administrators can submit jobs to this queue (unless FAIRSHARE is also defined), or switch any user's jobs into this queue, even if they are not listed.

Default `all`

SEE ALSO

lsf.cluster(5), lsf.conf(5), lsb.params(5), lsb.hosts(5), lsb.users(5), lsf.sudoers(5), bhpart(1), busers(1), bugroup(1), bchkpnt(1), nice(1), getgrnam(3), getrlimit(2), bmgroup(1), bqueues(1), bhosts(1), bsub(1), lsid(1), mbatchd(8), badmin(8)

lsb.users

Overview

The `lsb.users` file is used to configure user groups, hierarchical fairshare for users and user groups, and job slot limits for users and user groups. It is also used to configure account mappings in a MultiCluster environment.

This file is optional.

The `lsb.users` file is stored in the directory `LSB_CONFDIR/`*`cluster_name`*`/configdir`, where LSB_CONFDIR is defined in `lsf.conf`.

Contents

UserGroup Section

Optional. Defines user groups.

The name of the user group can be used in other user group and queue definitions, as well as on the command line. Specifying the name of a user group has exactly the same effect as listing the names of all users in the group.

The total number of user groups cannot be more than MAX_GROUPS in `lsbatch.h`.

Structure

The first line consists of two mandatory keywords, `GROUP_NAME` and `GROUP_MEMBER`. The `USER_SHARES` keyword is optional. Subsequent lines name a group and list its membership and optionally its share assignments.

Each line must contain one entry for each keyword. Use empty parentheses `()` or a dash `-` to specify the default value for an entry.

Example of a UserGroup Section

```
Begin UserGroup
GROUP_NAME   GROUP_MEMBER
groupA       (user1 user2 user3 user4)
groupB       (groupA user5)
groupC       (!)
End UserGroup
```

```
Begin UserGroup
GROUP_NAME   GROUP_MEMBER            USER_SHARES
groupB       (user1 user2)           ()
groupC       (user3 user4)           ([User3,3] [User4,4])
groupA       (GroupB GroupC User4)   ([User4,1] [default,10])
End UserGroup
```

GROUP_NAME

An alphanumeric string representing the user group name. You cannot use the reserved name `all` or a `/` in a group name, and group names must not conflict with user names.

GROUP_MEMBER

A list of user names or user group names that belong to the group, enclosed in parentheses and separated by spaces. Group names must not conflict with user names.

User and user group names can appear on multiple lines, because users can belong to multiple groups.

User groups may be defined recursively but must not create a loop.

Syntax **(***user_name* | *user_group* ...**)** | **(all)** | **(!)**

Specify the following, all enclosed in parentheses:

- *user_name* | *user_group*

  User and user group names, separated by spaces. User names must be valid login names.

  User group names can be LSF user groups defined previously in this section, or UNIX and Windows NT user groups.
- **all**

  The reserved name `all` specifies all users in the cluster.
- **!**

  The exclamation mark `!` specifies that the group membership should be retrieved via `egroup`.

USER_SHARES

Optional. Enables hierarchical fairshare and defines a share tree for users and user groups.

By default, when resources are assigned collectively to a group, the group members compete for the resources according to FCFS scheduling. You can use hierarchical fairshare to further divide the shares among the group members.

Syntax **([***user***,** *number_shares***])**

Specify the arguments as follows:

- Enclose the list in parentheses, even if you do not specify any user share assignments.
- Enclose each user share assignment in square brackets, as shown.

- Separate the list of share assignments with a space.
- *user*

  Specify users or user groups. You can assign the shares to:

  - A single user (specify *user_name*)
  - Users in a group, individually (specify *group_name*`@`) or collectively (specify *group_name*)
  - Users not included in any other share assignment, individually (specify the keyword `default`) or collectively (specify the keyword `others`)

  By default, when resources are assigned collectively to a group, the group members compete for the resources on a first-come, first-served (FCFS) basis. You can use hierarchical fairshare to further divide the shares among the group members.

  When resources are assigned to members of a group individually, the share assignment is recursive. Members of the group and of all subgroups always compete for the resources according to FCFS scheduling, regardless of hierarchical fairshare policies.

- *number_shares*

  Specify a positive integer representing the number of shares of the cluster resources assigned to the user.

  The number of shares assigned to each user is only meaningful when you compare it to the shares assigned to other users or to the total number of shares. The total number of shares is just the sum of all the shares assigned in each share assignment.

User Section

Optional. If this section is not defined, all users and user groups can run an unlimited number of jobs in the cluster.

This section defines the maximum number of jobs a user or user group can run concurrently in the cluster. This is to avoid situations in which a user occupies all or most of the system resources while other users' jobs are waiting.

Structure

All three fields are mandatory: `USER_NAME`, `MAX_JOBS`, `JL/P`.

You must specify a dash (`-`) to indicate the default value (unlimited) if a user or user group is specified. Fields cannot be left blank.

Example of a User Section

```
Begin User
USER_NAME   MAX_JOBS   JL/P
user1       10          -
user2        4          1
user3        -          2
groupA@     10          1
default      6          1
End User
```

USER_NAME

User or user group for which job slot limits are defined.

Use the reserved user name `default` to specify a job slot limit that applies to each user and user group not explicitly named. Since the limit specified with the keyword `default` applies to user groups also, ensure you select a limit that is high enough, or explicitly define limits for user groups.

User group names can be the LSF user groups defined previously, and/or UNIX and Windows NT user groups.

Job slot limits apply to a group as a whole. Append `@` to a group name to make the job slot limits apply individually to each user in the group. If a group contains a subgroup, the job slot limit also applies to each member in the subgroup recursively.

MAX_JOBS

Per-user or per-group job slot limit for the cluster. Total number of job slots that each user or user group can use in the cluster.

JL/P

Per processor job slot limit per user or user group.

Total number of job slots that each user or user group can use per processor. This job slot limit is configured per processor so that multiprocessor hosts will automatically run more jobs.

This number can be a fraction such as 0.5, so that it can also serve as a per-host limit. This number is rounded up to the nearest integer equal to or greater than the total job slot limits for a host. For example, if `JL/P` is 0.5, on a 4-CPU multiprocessor host, the user can only use up to 2 job slots at any time. On a uniprocessor machine, the user can use 1 job slot.

UserMap Section

Optional. Used only in an LSF MultiCluster environment.

Defines system-level account mapping for users and user groups.

To support the execution of batch jobs across non-uniform user name spaces between clusters, LSF allows user account mapping. For a job submitted by one user account in one cluster to run under a different user account in a remote cluster, both the local and remote clusters must have the account mapping properly configured.

Structure

All three fields `LOCAL`, `REMOTE` and `DIRECTION` are required.

Example of a UserMap Section

On cluster1

```
Begin UserMap
LOCAL    REMOTE                              DIRECTION
user1    user2@cluster2                      export
user3    (user4@cluster2 user6@cluster2)     export
End UserMap
```

On cluster2

```
Begin UserMap
LOCAL            REMOTE                  DIRECTION
user2            user1@cluster1          import
(user6 user8)    user3@cluster1          import
End UserMap
```

`Cluster1` configures `user1` to run jobs as `user2` and `user3` to run jobs as `user4` or `user6`.

`Cluster2` configures `user1` to run jobs as `user2` and `user3` to run jobs as `user6` or `user8`.

Only mappings configured in both clusters will work. The common account mappings are for `user1` to run jobs as `user2` and for `user3` to run jobs as `user6`. Therefore, these mappings will work, but the mappings of `user3` to `user4` and `user8` are only half-done and so will not work.

LOCAL

A list of users or user groups in the local cluster.

Multiple user names and user group names must be separated by a space, and the entire list enclosed in parentheses `()`.

REMOTE

A list of remote users or user groups in the form:
`user_name@cluster_name`
`user_group_name@cluster_name`

Multiple user names and user group names must be separated by a space, and the entire list enclosed in parentheses `()`.

DIRECTION

Configures the direction of account mapping:

- The `export` keyword configures local users/groups to run jobs as remote users/groups.
- The `import` keyword configures remote users/groups to run jobs as local users/groups.

Both directions must be configured for a mapping to work. The mapping must be configured in both the local and remote clusters.

SEE ALSO

lsf.cluster(5), lsf.conf(5), lsb.params(5), lsb.hosts(5), lsb.queues(5), bhosts(1), bmgroup(1), bhpart(1), busers(1), bugroup(1), bqueues(1), bsub(1), bchkpnt(1), lsid(1), nice(1), getgrnam(3), mbatchd(8), badmin(8)

lsf.cluster

Overview This is the cluster configuration file. There is one for each cluster, called `lsf.cluster.`*`cluster_name`*. The *cluster_name* suffix is the cluster name defined in the Cluster section of `lsf.shared`.

Contents

NewIndex Section

The NewIndex section in the `lsf.shared` file is obsolete. To achieve the same effect, the Resource section of the `lsf.shared` file can be used to define a dynamic numeric resource, and the `default` keyword can be used in the LOCATION field of the ResourceMap section of the `lsf.cluster.`*`cluster_name`* file.

Parameters Section

(Optional) This section contains miscellaneous parameters for the LIM.

ADJUST_DURATION

Syntax **ADJUST_DURATION =** *integer*

Description Integer reflecting a multiple of EXINTERVAL that controls the time period during which load adjustment is in effect.

The `lsplace(1)` and `lsloadadj(1)` commands artificially raise the load on a selected host. This increase in load decays linearly to 0 over time.

Default 3

ELIM_POLL_INTERVAL

Syntax **ELIM_POLL_INTERVAL =** *time_in_seconds*

Description Time interval, in seconds, in which the LIM daemon samples load information

This parameter only needs to be set if an ELIM is being used to report information more frequently than every 5 seconds.

Default 5 seconds

ELIMARGS

Syntax **ELIMARGS =** *cmd_line_args*

Description Specifies any necessary command-line arguments for the external LIM on startup

This parameter is ignored if no external load indices are configured.

Default None

EXINTERVAL

Syntax **EXINTERVAL =** *time_in_seconds*

Description Time interval, in seconds, at which the LIM daemons exchange load information

On extremely busy hosts or networks, or in clusters with a large number of hosts, load may interfere with the periodic communication between LIM daemons. Setting EXINTERVAL to a longer interval can reduce network load and slightly improve reliability, at the cost of slower reaction to dynamic load changes.

Default 15 seconds

FLOAT_CLIENTS

Syntax **FLOAT_CLIENTS =** *number_of_floating_client_licenses*

Description Sets the size of your license pool in the cluster

When the master LIM starts up, up to *number_of_floating_client_licenses* will be checked out for use as floating client licenses. If fewer licenses are available than specified by *number_of_floating_client_licenses*, only the available licenses will be checked out and used.

If FLOAT_CLIENTS is not specified in `lsf.cluster.`*`cluster_name`* or there is an error in either `license.dat` or in `lsf.cluster.`*`cluster_name`*, the floating LSF client license feature is disabled.

Warning **When the LSF floating client feature is enabled, any host will be able to submit jobs to the cluster. You can limit which hosts can be LSF floating clients with the parameter FLOAT_CLIENTS_ADDR_RANGE in `lsf.cluster.`*`cluster_name`*.**

Note **Although LSF Floating Client requires a license, LSF_Float_Client does not need to be added to the PRODUCTS line. LSF_Float_Client also cannot be added as a resource for specific hosts already defined in `lsf.cluster.`*`cluster_name`*. Should these lines be present, they are ignored by LSF.**

Default Undefined

FLOAT_CLIENTS_ADDR_RANGE

Syntax **`FLOAT_CLIENTS_ADDR_RANGE =`** *IP_address* ...

Description Optional. IP address or range of addresses, in dotted quad notation (`nnn.nnn.nnn.nnn`), of domains from which floating client hosts can submit requests. Multiple ranges can be defined, separated by spaces.

If the value of this parameter is undefined, there is no security and any host can be an LSF floating client.

If a value is defined, security is enabled. If there is an error in the configuration of this variable, by default, no host will be allowed to be an LSF floating client.

When this parameter is defined, client hosts that do not belong to the domain will be denied access.

If a requesting host belongs to an IP address that falls in the specified range, the host will be accepted to become an LSF floating client.

IP addresses are separated by spaces, and considered "OR" alternatives.

The asterisk (*) character indicates any value is allowed.

The dash (-) character indicates an explicit range of values. For example 1-4 indicates 1,2,3,4 are allowed.

Open ranges such as *-30, or 10-*, are allowed.

If a range is specified with less fields than an IP address such as 10.161, it is considered as 10.161.*.*.

Address ranges are validated at configuration time so they must conform to the required format. If any address range is not in the correct format, no host will be accepted as an LSF floating client and a error message will be logged in the LIM log.

This parameter is limited to 255 characters.

Notes After you configure FLOAT_CLIENTS_ADDR_RANGE, check the `lim.log.`*`host_name`* file to make sure this parameter is correctly set. If this parameter is not set or is wrong, this will be indicated in the log file.

Examples

- `FLOAT_CLIENTS_ADDR_RANGE=100`

  All client hosts with a domain address starting with 100 will be allowed access.

- `FLOAT_CLIENTS_ADDR_RANGE=100-110.34.1-10.4-56`

  All client hosts belonging to a domain with an address having the first number between 100 and 110, then 34, then a number between 1 and 10, then, a number between 4 and 56 will be allowed access. Example: 100.34.9.45, 100.34.1.4, 102.34.3.20, etc.

- `FLOAT_CLIENTS_ADDR_RANGE=100.172.1.13 100.*.30-54 124.24-*.1.*-34`

  All client hosts belonging to a domain with the address 100.172.1.13 will be allowed access. All client hosts belonging to domains starting with 100, then any number, then a range of 30 to 54 will be allowed access. All client hosts belonging to domains starting with 124, then from 24 onward, then 1, then from 0 to 34 will be allowed access.

- `FLOAT_CLIENTS_ADDR_RANGE=12.23.45.*`

  All client hosts belonging to domains starting with 12.23.45 are allowed.

- `FLOAT_CLIENTS_ADDR_RANGE=100.*43`

  The * character can only be used to indicate any value. In this example, an error will be inserted in the LIM log and no hosts will be accepted to become LSF floating clients.

- `FLOAT_CLIENTS_ADDR_RANGE=100.*43 100.172.1.13`

Although one correct address range is specified, because *43 is incorrect format, the entire line is considered invalid. An error will be inserted in the LIM log and no hosts will be accepted to become LSF floating clients.

Default Undefined. No security is enabled. Any host in any domain is allowed access to LSF floating client licenses.

HOST_INACTIVITY_LIMIT

Syntax **HOST_INACTIVITY_LIMIT =** *integer*

Description Integer reflecting a multiple of EXINTERVAL that controls the maximum time a slave LIM will take to send its load information to the master LIM as well as the frequency at which the master LIM will send a heartbeat message to its slaves.

A slave LIM can send its load information any time from EXINTERVAL to (HOST_INACTIVITY_LIMIT-2)*EXINTERVAL seconds. A master LIM will send a master announce to each host at least every EXINTERVAL*HOST_INACTIVITY_LIMIT seconds.

Default 5

LSF_ELIM_BLOCKTIME

Syntax **LSF_ELIM_BLOCKTIME=***seconds*

Description UNIX only.

Maximum amount of time LIM waits for a load update string from the ELIM if it is not immediately available.

Use this parameter to add fault-tolerance to LIM when using ELIMs. If there is an error in the ELIM or some situation arises that the ELIM cannot send the entire load update string to the LIM, LIM will not wait indefinitely for load information from ELIM. After the time period specified by LSF_ELIM_BLOCKTIME, the LIM writes the last string sent by ELIM in its log file (`lim.log.`*`host_name`*) and restarts the ELIM.

For example, if LIM is expecting 3 name-value-pairs, such as:

```
3 tmp2 49.5 nio 367.0 licenses 3
```

If after the time period specified by LSF_ELIM_BLOCKTIME LIM has only received the following:

```
3 tmp2 47.5
```

LIM writes whatever was received last (`3 tmp2 47.5`) in the log file and restarts the ELIM.

Valid Values Non-negative integers

A value of 0 indicates that LIM will not wait at all to receive information from ELIM—it expects to receive the entire load string at once.

So, if for example, your ELIM writes value-pairs with 1 second intervals between them, and you collect 12 load indices, you need to allow at least 12 seconds for the ELIM to complete writing an entire load string. So you would define LSF_ELIM_BLOCKTIME to 15 or 20 seconds for example.

Default Undefined—LIM waits indefinitely to receive load information from ELIM

LSF_ELIM_DEBUG

Syntax **`LSF_ELIM_DEBUG=y`**

Description UNIX only.

This parameter is useful to view which load information an ELIM is collecting and to add fault-tolerance to LIM.

When this parameter is set to `y`:

- All load information received by LIM from the ELIM is logged in the LIM log file (`lim.log.`*`host_name`*).
- If LSF_ELIM_BLOCKTIME is undefined, whenever there is an error in the ELIM or some situation arises that the ELIM cannot send the entire load update string to the LIM, LIM does not wait indefinitely for load information from ELIM. After 2 seconds, the LIM restarts the ELIM.

For example, LIM is expecting 3 name-value-pairs, such as:

```
3 tmp2 47.5 nio 344.0 licenses 5
```

However, LIM only receives the following from ELIM:

```
3 tmp2 47.5
```

LIM waits 2 seconds after the last value is received and if no more information is received, LIM restarts the ELIM.

If LSF_ELIM_BLOCKTIME is defined, the LIM waits for the specified amount of time before restarting the ELIM instead of the 2 seconds.

Default Undefined—if LSF_ELIM_DEBUG is undefined, load information sent from ELIM to LIM is not logged. In addition, if LSF_ELIM_BLOCKTIME is undefined, LIM waits indefinitely to receive load information from ELIM.

See Also LSF_ELIM_BLOCKTIME to configure how long LIM waits before restarting the ELIM, use the parameter LSF_ELIM_BLOCKTIME.

LSF_ELIM_RESTARTS to limit how many times the ELIM can be restarted

LSF_ELIM_RESTARTS

Syntax **LSF_ELIM_RESTARTS=***integer*

Description UNIX only.

LSF_ELIM_BLOCKTIME or LSF_ELIM_DEBUG must be defined in conjunction with LSF_ELIM_RESTARTS.

Defines the maximum number of times an ELIM can be restarted.

When this parameter is defined:

- If LIM attempts to retrieve load information from the ELIM and there is an error such as an invalid value for example, LIM restarts the ELIM.

If the error is consistent and LIM keeps restarting the ELIM, LSF_ELIM_RESTARTS limits how many times the ELIM can be restarted to prevent an ongoing loop.

Valid Values Non-negative integers

Default Undefined; the number of ELIM restarts is unlimited

See Also LSF_ELIM_BLOCKTIME, LSF_ELIM_DEBUG

MASTER_INACTIVITY_LIMIT

Syntax **MASTER_INACTIVITY_LIMIT =** *integer*

Description An integer reflecting a multiple of EXINTERVAL. A slave will attempt to become master if it does not hear from the previous master after (HOST_INACTIVITY_LIMIT +*host_number**MASTER_INACTIVITY_LIMIT)*EXINTERVAL seconds, where *host_number* is the position of the host in `lsf.cluster.cluster_name`.

The master host is *host_number* 0.

Default 2

PROBE_TIMEOUT

Syntax **PROBE_TIMEOUT =** *time_in_seconds*

Description Specifies the timeout in seconds to be used for the `connect`(2) system call

Before taking over as the master, a slave LIM will try to connect to the last known master via TCP.

Default 2 seconds

PRODUCTS

Syntax **PRODUCTS =** *product_name ...*

Description Enables specified LSF products for all hosts in the cluster, subject to license availability

The PRODUCTS parameter is set automatically by the `lsfsetup` installation script during license installation.

Specify a space-separated list of LSF Suite products:

- LSF_Base
- LSF_Batch
- LSF_JobScheduler
- LSF_MultiCluster
- LSF_Analyzer
- LSF_Make
- LSF_Parallel
- LSF_Resource_Preempt

LSF_Base is automatically enabled if LSF_Batch, LSF_JobScheduler, or LSF_MultiCluster is enabled.

Individual hosts can be configured to run as LSF Batch servers or LSF JobScheduler servers within the same cluster. LSF MultiCluster is either enabled or disabled for MultiCluster operation for the entire cluster.

If you specify products without having a valid license file, the host will be unlicensed only for those products. The cluster will remain operational for the remaining products for which licenses are available.

Default LSF_Base LSF_Batch

RETRY_LIMIT

Syntax **RETRY_LIMIT =** *integer*

Description Integer reflecting a multiple of EXINTERVAL that controls the number of retries a master or slave LIM makes before assuming that the slave or master is unavailable.

If the master does not hear from a slave for HOST_INACTIVITY_LIMIT exchange intervals, it will actively poll the slave for RETRY_LIMIT exchange intervals before it will declare the slave as unavailable. If a slave does not hear from the master for HOST_INACTIVITY_LIMIT exchange intervals, it will actively poll the master for RETRY_LIMIT intervals before assuming that the master is down.

Default 2

ClusterAdmins Section

(Optional) The `ClusterAdmins` section defines the LSF administrators for the cluster. The only keyword is ADMINISTRATORS.

If the `ClusterAdmins` section is not present, the default LSF administrator is `root`. Using `root` as the primary LSF administrator is not recommended.

ADMINISTRATORS

Syntax **`ADMINISTRATORS =`** *administrator_name ...*

Description Specify UNIX and Windows user and user group names.

On Windows NT:

Windows NT domain

- If the specified user or user group is a domain administrator, member of the `Power Users` group or a group with domain administrative privileges, the specified user or user group must belong to the LSF user domain.
- If the specified user or user group is a user or user group with a lower degree of privileges than outlined in the previous point, the user or user group must belong to the LSF user domain and be part of the `Global Admins` group.

Windows NT workgroup

- If the specified user or user group is not a workgroup administrator, member of the `Power Users` group, or a group with administrative privileges on each host, the specified user or user group must belong to the `Local Admins` group on each host.

The first administrator of the expanded list is considered the primary LSF administrator. The primary administrator is the owner of the LSF configuration files, as well as the working files under `LSB_SHAREDIR/`*`cluster_name`*. If the primary administrator is changed, make sure the owner of the configuration files and the files under `LSB_SHAREDIR/`*`cluster_name`* are changed as well.

Administrators other than the primary LSF administrator have the same privileges as the primary LSF administrator except that they do not have permission to change LSF configuration files. They can perform clusterwide operations on jobs, queues, or hosts in the system.

For flexibility, each cluster may have its own LSF administrators, identified by a user name, although the same administrators can be responsible for several clusters.

Use the `-l` option of the `lsclusters(1)` command to display all of the administrators within a cluster.

Compatibility

For backwards compatibility, ClusterManager and Manager are synonyms for `ClusterAdmins` and ADMINISTRATORS respectively. It is possible to have both sections present in the same `lsf.cluster.`*`cluster_name`* file to allow daemons from different LSF versions to share the same file.

Default

`lsfadmin`

Example

The following gives an example of a cluster with three LSF administrators. The user listed first, `user2`, is the primary administrator. The user group `lsfgrp` and the user `user7` are secondary administrators.

```
Begin ClusterAdmins
ADMINISTRATORS = user2 lsfgrp user7
End ClusterAdmins
```

Host Section

The Host section is the last section in `lsf.cluster.`*`cluster_name`* and is the only required section. It lists all the hosts in the cluster and gives configuration information for each host.

The order in which the hosts are listed in this section is important. The LIM on the first host listed becomes the master LIM if this host is up; otherwise, that on the second becomes the master if its host is up, and so on.

Since the master LIM makes all placement decisions for the cluster, it should be on a fast machine. Also, to avoid the delays involved in switching masters if the first machine goes down, the master should be on a reliable machine. It is desirable to arrange the list such that the first few hosts in the list are always in the same subnet. This avoids a situation where the second host takes over as master when there are communication problems between subnets.

Configuration information is of two types:

- Some fields in a host entry simply describe the machine and its configuration.
- Other fields set thresholds for various resources.

Descriptive Fields

The following fields are required in the Host section:

- HOSTNAME
- RESOURCES
- type
- server

The following fields are optional:

- model
- nd
- RUNWINDOW
- REXPRI

HOSTNAME

Description Official name of the host as returned by `hostname`(1)

The name must be listed in `lsf.shared` as belonging to this cluster.

model

Description Host model

The name must be defined in the HostModel secton of `lsf.shared`. This determines the CPU speed scaling factor applied in load and placement calculations.

If you leave the model or type column blank or enter the ! keyword, you are indicating that the host model or type is to be automatically detected by the LIM running on the host.

nd

Description Number of local disks

This corresponds to the ndisks static resource. On most host types, LSF automatically determines the number of disks, and the nd parameter is ignored.

nd should only count local disks with file systems on them. Do not count either disks used only for swapping or disks mounted with NFS.

Default The number of disks determined by the LIM, or 1 if the LIM cannot determine this

RESOURCES

Description The static Boolean resources available on this host

The resource names are strings defined in the Resource section of `lsf.shared`. You may list any number of resources, enclosed in parentheses and separated by blanks or tabs: for example:

```
(fs frame hpux)
```

Optionally, you can specify a dedicated resource by prefixing the resource with an exclamation mark (!). A host with dedicated resources is not selected by LIM for a job unless a dedicated resource name is explicitly specified in the resource requirements for the job.

REXPRI

Description (UNIX ONLY) Default execution priority for interactive remote jobs run under the RES

The range is from -20 to 20. REXPRI corresponds to the BSD-style nice value used for remote jobs. For hosts with System V-style nice values with the range 0 - 39, a REXPRI of -20 corresponds to a nice value of 0, and +20 corresponds to 39. Higher values of REXPRI correspond to lower execution priority; -20 gives the highest priority, 0 is the default priority for login sessions, and +20 is the lowest priority.

Default 0

RUNWINDOW

Description Dispatch window during this host is accepts remote interactive tasks

When the host is not available for remote execution, the host status is `lockW` (locked by run window). LIM does not schedule interactive tasks on hosts locked by dispatch windows. Note that run windows only apply to interactive tasks placed by LIM. LSF Batch uses its own (optional) host dispatch windows to control batch job processing on batch server hosts.

Format A dispatch window consists of one or more time windows in the format *begin_time-end_time*. No blanks can separate *begin_time* and *end_time*. Time is specified in the form [*day*:]*hour*[:*minute*]. If only one field is specified, LSF assumes it is an *hour*. Two fields are assumed to be *hour*:*minute*. Use blanks to separate time windows.

Default Always accept remote jobs

server

Description Indicates whether the host can receive jobs from other hosts

Specify 1 if the host can receive jobs from other hosts; specify 0 otherwise. If server is set to 0, the host is an LSF client. Client hosts do not run the LSF daemons. Client hosts can submit interactive and batch jobs to an LSF cluster, but they cannot execute jobs sent from other hosts.

Default 1

type

Description Host type as defined in the HostType section of `lsf.shared`

The strings used for host types are determined by the system administrator: for example, SUNSOL, DEC, or HPPA. The host type is used to identify binary-compatible hosts.

The host type is used as the default resource requirement. That is, if no resource requirement is specified in a placement request, the task is run on a host of the same type as the sending host.

Often one host type can be used for many machine models. For example, the host type name SUNSOL6 might be used for any computer with a SPARC processor running SunOS 6. This would include many Sun models and quite a few from other vendors as well.

If you leave the model or type column blank or enter the ! keyword, you are indicating that the host model or type is to be automatically detected by the LIM running on the host.

Threshold Fields

The LIM uses these thresholds in determining whether to place remote jobs on a host. If one or more LSF load indices exceeds the corresponding threshold (too many users, not enough swap space, etc.), then the host is regarded as busy, and LIM will not recommend jobs to that host.

The CPU run queue length threshold values (r15s, r1m, and r15m) are taken as effective queue lengths as reported by `lsload -E`.

All of these fields are optional; you only need to configure thresholds for load indices that you wish to use for determining whether hosts are busy. Fields that are not configured are not considered when determining host status. The keywords for the threshold fields are not case sensitive.

Thresholds can be set for any of the following:

- The built-in LSF load indexes (r15s, r1m, r15m, ut, pg, it, io, ls, swp, mem, tmp)
- External load indexes defined in the Resource section of `lsf.shared`

Example of a Host Section

This example Host section contains descriptive and threshold information for two hosts:

```
Begin Host
HOSTNAME   model    type   server r1m pg tmp RESOURCES     RUNWINDOW
hostA      SparcIPC Sparc  1      3.5 15   0 (sunos frame  ()
hostD      Sparc10  Sparc  1      3.5 15   0 (sunos)       (5:18:30-1:8:30)
End Host
```

ResourceMap Section

The ResourceMap section defines shared resources in your cluster. This section specifies the mapping between shared resources and their sharing hosts. When you define resources in the Resources section of `lsf.shared`, there is no distinction between a shared and non-shared resource. By default, all resources are not shared and are local to each host. By defining the ResourceMap section, you can define resources that are shared by all hosts in the cluster or define resources that are shared by only some of the hosts in the cluster.

This section must appear after the Host section of `lsf.cluster.`*`cluster_name`*, because it has a dependency on host names defined in the Host section. The following parameters must be defined in the ResourceMap section:

ResourceMap Section Structure

The first line consists of the keywords RESOURCENAME and LOCATION. Subsequent lines describe the hosts that are associated with each configured resource.

LOCATION

Description Defines the hosts that share the resource

For a static resource, you must define a value here as well. Do not define a value for a dynamic resource.

instance is a list of host names that share an instance of the resource. The reserved words `all`, `others`, and `default` can be specified for the instance:

- `all`—Indicates that there is only one instance of the resource in the whole cluster and that this resource is shared by all of the hosts

  Use the not operator (~) to exclude hosts from the `all` specification. For example:

  ```
  (2@[all ~host3 ~host4])
  ```

means that 2 units of the resource are shared by all server hosts in the cluster made up of host1 host2 ... host*n*, except for host3 and host4. This is useful if you have a large cluster but only want to exclude a few hosts.

The parentheses are required in the specification. The not operator can only be used with the `all` keyword. It is not valid with the keywords `others` and `default`.

- `others`—Indicates that the rest of the server hosts not explicitly listed in the LOCATION field comprise one instance of the resource

  For example:

  ```
  2@[host1] 4@[others]
  ```

  indicates that there are 2 units of the resource on apple and 4 units of the resource shared by all other hosts.

- `default`—Indicates an instance of a resource on each host in the cluster

  This specifies a special case where the resource is in effect not shared and is local to every host. `default` means at each host. Normally, you should not need to use `default`, because by default all resources are local to each host. You might want to use ResourceMap for a non-shared static resource if you need to specify different values for the resource on different hosts.

RESOURCENAME

Description Name of the resource

This resource name must be defined in the Resource section of `lsf.shared`. You must specify at least a name and description for the resource, using the keywords RESOURCENAME and DESCRIPTION.

- A resource name cannot begin with a number.
- A resource name cannot contain any of the following characters:

  `: . ( ) [ + - * / ! & | < > @ =`
- A resource name cannot be any of the following reserved names:

  `cpu cpuf io logins ls idle maxmem maxswp maxtmp type model status it mem ncpus ndisks pg r15m r15s r1m swap swp tmp ut`
- Resource names are case sensitive
- Resource names can be up to 29 characters in length

Example of a ResourceMap Section

```
Begin ResourceMap
RESOURCENAME   LOCATION
verilog        [5@all]
local          ([host1 host2] [others])
End ResourceMap
```

The resource `verilog` must already be defined in the RESOURCE section of the `lsf.shared` file. It is a static numeric resource shared by all hosts. The value for `verilog` is 5. The resource `local` is a numeric shared resource that contains two instances in the cluster. The first instance is shared by two machines, `host1` and `host2`. The second instance is shared by all other hosts.

Resources defined in the ResourceMap section can be viewed by using the `-s` option of the `lshosts` (for static resource) and `lsload` (for dynamic resource) commands.

RemoteClusters Section

(Optional) This section is used only in an LSF MultiCluster environment. By default, the local cluster can obtain information about all other clusters specified in `lsf.shared`. The RemoteClusters section limits the clusters that the local cluster can obtain information about.

The first line consists of keywords. CLUSTERNAME is mandatory and the other parameters are optional.

Subsequent lines configure the remote cluster.

CACHE_INTERVAL

Description (Optional) Specifies the length of time that load information from the remote cluster is cached, in seconds.

Host configuration information is cached twice as long as load information.

When a command requests load or host information, it is retrieved from the relevant remote cluster and is also cached by the local master LIM. The next time a command requests the information, clients in the local cluster receive the cached copy of the remote cluster information if the cache has not expired.

Upon a request from an LSLIB call or an LSF command, the cached information is used if it is less than CACHE_INTERVAL old. If the cached information is older than CACHE_INTERVAL, fresh information is retrieved from the remote cluster by the local master LIM.

Default 60 (seconds)

CLUSTERNAME

Description Remote cluster name

CLUSTERNAME causes the local cluster to be aware of the remote cluster.

EQUIV

Description (Optional) Indicator of whether to make the remote cluster equivalent to the local cluster

Specify 'Y' to make the remote cluster equivalent to the local cluster. Otherwise, specify 'N'. The master LIM considers all equivalent clusters when servicing requests from clients for load, host, or placement information.

EQUIV changes the default behavior of LSF commands and utilities and causes them to automatically return load (`lsload(1)`), host (`lshosts(1)`), or placement (`lsplace(1)`) information about the remote cluster as well as the local cluster, even when you don't specify a cluster name.

Example

```
CLUSTERNAME  EQUIV   CACHE_INTERVAL  RECV_FROM
cluster1       Y           60            Y
```

RECV_FROM

Description (Optional) Specifies whether interactive tasks from the remote cluster will be accepted by the local cluster

RECV_FROM does not affect regular LSF Batch jobs or interactive Batch jobs.

Specify 'N' if you want the local cluster to reject interactive tasks from the remote cluster. Otherwise, specify 'Y'.

Example+

```
CLUSTERNAME  EQUIV   CACHE_INTERVAL  RECV_FROM
cluster1       N           60            N
```

Default Y

lsf.conf

Overview Installation of and operation of LSF is controlled by the `lsf.conf` file. This chapter explains the contents of the `lsf.conf` file.

Contents

About lsf.conf

The `lsf.conf` file is created during installation by the LSF setup program, and records all the settings chosen when LSF was installed. The `lsf.conf` file dictates the location of the specific configuration files and operation of individual servers and applications.

The `lsf.conf` file is used by LSF and applications built on top of it. For example, information in `lsf.conf` is used by LSF daemons and commands to locate other configuration files, executables, and network services. `lsf.conf` is updated, if necessary, when you upgrade to a new version.

This file can also be expanded to include LSF application-specific parameters.

Location

The default location of `lsf.conf` is in `/etc`. This default location can be overridden when necessary by either the environment variable LSF_ENVDIR or the command line option `-d` available to some of the applications.

Format

Each entry in `lsf.conf` has one of the following forms:

```
NAME=VALUE
```

```
NAME=
```

```
NAME= "STRING1 STRING2 ..."
```

The equal sign = must follow each `NAME` even if no value follows and there should be no space beside the equal sign.

A value that contains multiple strings separated by spaces must be enclosed in quotation marks.

Lines starting with a pound sign (#) are comments and are ignored.

Parameters

LSB_API_CONNTIMEOUT

Syntax **LSB_API_CONNTIMEOUT =** *time_seconds*

Description The timeout in seconds when connecting to the Batch system.

Valid Values Any positive integer or zero

Default 10

See Also LSB_API_RECVTIMEOUT

LSB_API_RECVTIMEOUT

Syntax **LSB_API_RECVTIMEOUT =** *time_seconds*

Description Timeout in seconds when waiting for a reply from the Batch system.

Valid Values Any positive integer or zero

Default 0

See Also LSB_API_CONNTIMEOUT

LSB_CMD_LOG_MASK

Syntax **LSB_CMD_LOG_MASK =** *log_level*

Description Specifies the logging level of error messages from LSF Batch commands.

For example:

```
LSB_CMD_LOG_MASK=LOG_DEBUG
```

Batch commands log error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. The level specified by LSB_CMD_LOG_MASK determines which messages are recorded and which are discarded. All messages logged at the specified level or higher are recorded, while lower level messages are discarded.

For debugging purposes, the level LOG_DEBUG contains the fewest number of debugging messages and is used only for very basic debugging. The level LOG_DEBUG3 records all debugging messages, and is not often used. Most debugging is done at the level LOG_DEBUG2.

The commands log to the `syslog` facility unless LSB_CMD_LOGDIR is set.

Valid Values The log levels from highest to lowest are:

- LOG_EMERG
- LOG_ALERT
- LOG_CRIT
- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

Default LOG_WARNING

See Also LSB_CMD_LOGDIR, LSB_DEBUG, LSB_TIME_CMD, LSF_LOG_MASK, LSB_DEBUG_CMD, LSF_TIME_CMD

LSB_CMD_LOGDIR

Syntax **LSB_CMD_LOGDIR =** *path*

Description Specifies the path to the Batch command log files.

Default `/tmp`

See Also LSB_CMD_LOG_MASK, LSF_LOGDIR

LSB_CONFDIR

Syntax **LSB_CONFDIR =** *dir*

Description Specifies the path to the directory containing the LSF configuration files.

LSF Batch and LSF JobScheduler configuration directories are installed under LSB_CONFDIR.

Configuration files for each LSF cluster are stored in a subdirectory of LSB_CONFDIR. This subdirectory contains several files that define LSF Batch user and host lists, operation parameters, and queues.

All files and directories under LSB_CONFDIR must be readable from all hosts in the cluster. `LSB_CONFDIR/cluster_name/configdir` must be owned by the LSF administrator.

Do not redefine this parameter after LSF has been installed. To move these directories to another location, use `lsfsetup` and choose the `Product Install` option to install configuration files in another location.

This parameter also applies to JobScheduler configuration files.

Default `LSF_CONFDIR/lsbatch`

See Also LSF_CONFDIR

LSB_CRDIR

Syntax **LSB_CRDIR =** *dir*

Description Specifies the path and directory to the checkpointing executables on systems that support kernel-level checkpointing. LSB_CRDIR specifies the directory containing the `chkpnt` and `restart` utility programs that SBD uses to checkpoint or restart a job.

For example:

```
LSB_CRDIR=/usr/bin
```

If your platform supports kernel-level checkpointing, and if you want to use the utility programs provided for kernel-level checkpointing, set LSB_CRDIR to the location of the utility programs.

Default Undefined

If undefined, the system uses `/bin`.

LSB_DEBUG

Syntax **LSB_DEBUG = 1 | 2**

Description Sets the Batch system to debug.

If defined, LSF Batch will run in single user mode. In this mode, no security checking is performed; do not run the LSF Batch daemons as `root`.

When LSB_DEBUG is defined, LSF will not look in the system services database for port numbers. Instead, it uses the port numbers defined by the parameters LSB_MBD_PORT/LSB_SBD_PORT in `lsf.conf`. If these parameters are not defined, it uses port number 40000 for MBD and port number 40001 for SBD.

You should always specify 1 for this parameter unless you are testing LSF Batch.

Can also be defined from the command line.

Valid Values

- LSB_DEBUG=1

LSF Batch runs in the background with no associated control terminal.

- LSB_DEBUG=2
  LSF Batch runs in the foreground and prints error messages to `tty`.

Default Undefined

See Also LSF_LIM_DEBUG, LSF_RES_DEBUG

LSB_DEBUG_CMD

Syntax **LSB_DEBUG_CMD =** *log_class*

Description Sets the debugging log class for commands and APIs.

Specifies the log class filtering that will be applied to Batch commands or the API. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_CMD (which sets the log class) is used in combination with LSB_CMD_LOG_MASK (which sets the log level). For example:

```
LSB_CMD_LOG_MASK=LOG_DEBUG
LSB_DEBUG_CMD="LC_TRACE LC_EXEC"
```

Debugging is turned on when you define both parameters.

The daemons log to the `syslog` facility unless LSB_CMD_LOGDIR is defined.

To specify multiple log classes, use a space-separated list enclosed by quotation marks. For example:

```
LSB_DEBUG_CMD="LC_TRACE LC_EXEC"
```

Can also be defined from the command line.

Valid Values Valid log classes are:

- LC_AFS - Log AFS messages
- LC_AUTH - Log authentication messages
- LC_CHKPNT - Log checkpointing messages

- LC_COMM - Log communication messages
- LC_DCE - Log messages pertaining to DCE support
- LC_EEVENTD - Log eeventd messages
- LC_ELIM - Log ELIM messages
- LC_EXEC - Log significant steps for job execution
- LC_FAIR - Log fairshare policy messages
- LC_FILE - Log file transfer messages
- LC_HANG - Mark where a program might hang
- LC_JARRAY - Log job array messages
- LC_JGRP - Log job group messages
- LC_JLIMIT - Log job slot limit messages
- LC_LICENCE - Log license management messages
- LC_LOADINDX - Log load index messages
- LC_M_LOG - Log multievent logging messages
- LC_MPI - Log MPI messages
- LC_MULTI - Log messages pertaining to MultiCluster
- LC_PEND - Log messages related to job pending reasons
- LC_PERFM - Log performance messages
- LC_PIM - Log PIM messages
- LC_PREEMPT - Log preemption policy messages
- LC_SCHED - Log JobScheduler messages
- LC_SIGNAL - Log messages pertaining to signals
- LC_SYS - Log system call messages
- LC_TRACE - Log significant program walk steps
- LC_XDR - Log everything transferred by XDR

Default Undefined

See Also LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR

LSB_DEBUG_MBD

Syntax **LSB_DEBUG_MBD =** *log_class*

Description Sets the debugging log class for MBD.

Specifies the log class filtering that will be applied to MBD. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_MBD (which sets the log class) is used in combination with LSF_LOG_MASK (which sets the log level). For example:

```
LSF_LOG_MASK=LOG_DEBUG
LSB_DEBUG_MBD="LC_TRACE LC_EXEC"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_MBD="LC_TRACE LC_EXEC"
```

You need to restart the daemons after setting LSB_DEBUG_MBD for your changes to take effect.

If you use the command `badmin mbddebug` to temporarily change this parameter without changing `lsf.conf`, you will not need to restart the daemons.

The daemons log to the `syslog` facility unless LSF_LOGDIR is defined.

Valid Values Valid log classes are the same as for LSB_DEBUG_CMD except for the log classes LC_ELIM and LC_JARRAY which cannot be used with LSB_DEBUG_MBD. See "LSB_DEBUG_CMD" on page 383.

Default Undefined

See Also LSF_LOG_MASK, LSF_LOGDIR, `badmin mbddebug`

LSB_DEBUG_NQS

Syntax **`LSB_DEBUG_NQS =`** *log_class*

Description Sets the log class for debugging the NQS interface.

Specifies the log class filtering that will be applied to NQS. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_NQS (which sets the log class) is used in combination with LSF_LOG_MASK (which sets the log level). For example:

```
LSF_LOG_MASK=LOG_DEBUG
LSB_DEBUG_NQS="LC_TRACE LC_EXEC"
```

Debugging is turned on when you define both parameters.

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_NQS="LC_TRACE LC_EXEC"
```

The daemons log to the `syslog` facility unless LSF_LOGDIR is defined.

This parameter can also be defined from the command line.

Valid Values For a list of valid log classes, see "LSB_DEBUG_CMD" on page 383.

Default Undefined

See Also LSB_DEBUG_CMD, LSF_LOG_MASK, LSF_LOGDIR

LSB_DEBUG_SBD

Syntax **LSB_DEBUG_SBD =** *log_class*

Description Sets the debugging log class for SBD.

Specifies the log class filtering that will be applied to SBD. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_SBD (which sets the log class) is used in combination with LSF_LOG_MASK (which sets the log level). For example:

```
LSF_LOG_MASK=LOG_DEBUG
LSB_DEBUG_SBD="LC_TRACE LC_EXEC"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_SBD="LC_TRACE LC_EXEC"
```

You need to restart the daemons after setting LSB_DEBUG_SBD for your changes to take effect.

If you use the command `badmin sbddebug` to temporarily change this parameter without changing `lsf.conf`, you will not need to restart the daemons.

The daemons log to the `syslog` facility unless LSF_LOGDIR is defined.

Valid Values Valid log classes are the same as for LSB_DEBUG_CMD except for the log classes LC_ELIM and LC_JARRAY which cannot be used with LSB_DEBUG_SBD. See "LSB_DEBUG_CMD" on page 383.

Default Undefined

See Also LSB_DEBUG_MBD, LSF_LOG_MASK, LSF_LOGDIR, `badmin`

LSB_ECHKPNT_METHOD

Syntax **`LSB_ECHKPNT_METHOD=`***`method_name`*

Description Name of custom `echkpnt` and `erestart` methods.

Can also be defined as an environment variable, or specified through the `bsub -k` option.

The name you specify here will be used for both your custom `echkpnt` and `erestart` programs. You must assign your custom ***`echkpnt`*** and ***`erestart`*** programs the name `echkpnt.`***`method_name`*** and `erestart.`***`method_name`***. The programs `echkpnt.`***`method_name`*** and `erestart.`***`method_name`***. must be in LSF_SERVERDIR or in the directory specified by LSB_ECHKPNT_METHOD_DIR.

Do not define `LSB_ECHKPNT_METHOD=default` as `default` is a reserved keyword to indicate to use LSF's default `echkpnt` and `erestart` methods. You can however, specify `bsub -k "my_dir method=default" my_job` to indicate that you want to use LSF's default checkpoint and restart methods.

When this parameter is undefined in `lsf.conf` or as an environment variable and no custom method is specified at job submission through `bsub -k`, LSF uses `echkpnt.default` and `erestart.default` to checkpoint and restart jobs.

When this parameter is defined, LSF uses the custom checkpoint and restart methods specified.

Limitations The method name and directory(LSB_ECHKPNT_METHOD_DIR) combination must be unique in the cluster.

For example, you may have two `echkpnt` applications with the same name such as `echkpnt.mymethod` but what differentiates them is the different directories defined with LSB_ECHKPNT_METHOD_DIR. It is the cluster administrator's responsibility to ensure that method name and method directory combinations are unique in the cluster.

Default Undefined; LSF uses `echkpnt.default` and `erestart.default` to checkpoint and restart jobs

Product LSF Base, LSF Batch

See Also LSB_ECHKPNT_METHOD_DIR, LSB_ECHKPNT_KEEP_OUTPUT

LSB_ECHKPNT_METHOD_DIR

Syntax **`LSB_ECHKPNT_METHOD_DIR=`***`path`*

Description Absolute path name of the directory in which custom `echkpnt` and `erestart` programs are located.

Can also be defined as an environment variable.

Default Undefined; LSF searches in LSF_SERVERDIR for custom `echkpnt` and `erestart` programs

Product LSF Base, LSF Batch

See Also LSB_ECHKPNT_METHOD, LSB_ECHKPNT_KEEP_OUTPUT

LSB_ECHKPNT_KEEP_OUTPUT

Syntax **`LSB_ECHKPNT_KEEP_OUTPUT=y|Y`**

Description Saves the standard output and standard error of custom echkpnt and erestart methods to:

- *`checkpoint_dir`*`/$LSB_JOBID/echkpnt.out`
- *`checkpoint_dir`*`/$LSB_JOBID/echkpnt.err`

- *checkpoint_dir*/$LSB_JOBID/erestart.out
- *checkpoint_dir*/$LSB_JOBID/erestart.err

Can also be defined as an environment variable.

Default Undefined; standard error and standard output messages from custom `echkpnt` and `erestart` programs is directed to `/dev/null` and discarded by LSF.

Product LSF Base, LSF Batch

See Also LSB_ECHKPNT_METHOD, LSB_ECHKPNT_METHOD_DIR

LSB_INTERACT_MSG_ENH

Syntax **`LSB_INTERACT_MSG_ENH = y | Y`**

Description If set, enables enhanced messaging for interactive batch jobs. To disable interactive batch job messages, set LSB_INTERACT_MSG_ENH to any value other than `y` or `Y`; for example, `LSB_INTERACT_MSG_ENH=N`.

Default Undefined

See Also LSB_INTERACT_MSG_INTVAL

LSB_INTERACT_MSG_INTVAL

Syntax **`LSB_INTERACT_MSG_INTVAL =`** *seconds*

Description Specifies the update interval in seconds for interactive batch job messages. LSB_INTERACT_MSG_INTVAL is ignored if LSB_INTERACT_MSG_ENH is not set.

Because the job information that LSF uses to get the pending or suspension reason is updated according to the value of MBD_SLEEP_TIME, there is no advantage to setting LSB_INTERACT_MSG_INTVAL less than MBD_SLEEP_TIME

Default Undefined. If LSB_INTERACT_MSG_INTVAL is set to an incorrect value, the default update interval is 60 seconds.

See Also LSB_INTERACT_MSG_ENH in `lsf.conf`, MBD_SLEEP_TIME in `lsb.params`

LSB_JOB_CPULIMIT

Syntax **`LSB_JOB_CPULIMIT = y | n`**

Description Determines whether the CPU limit is a per-process limit enforced by the OS or whether it is a per-job limit enforced by LSF:

- The per-process limit is enforced by the OS when the CPU time of one process of the job exceeds the CPU limit.
- The per-job limit is enforced by LSF when the total CPU time of all processes of the job exceed the CPU limit.

This parameter applies to CPU limits set when a job is submitted with `bsub -c` , and to CPU limits set for queues by CPULIMIT in `lsb.queues`.

The setting of LSB_JOB_CPULIMIT has the following effect on how the limit is enforced:

When LSB_JOB_CPULIMIT is	LSF-enforced per-job limit	OS-enforced per-process limit
y	Enabled	Disabled
n	Disabled	Enabled
undefined	Enabled	Enabled

- LSF-enforced per-job limit—When the sum of the CPU time of all processes of a job exceed the CPU limit, LSF sends a SIGXCPU signal (where supported by the operating system) from the operating system to all processes belonging to the job, then SIGINT, SIGTERM and SIGKILL. The interval between signals is 10 seconds by default.

  Note that SIGXCPU is not supported by Windows NT.

  On UNIX, the time interval between SIGXCPU, SIGINT, SIGKILL, SIGTERM can be configured with the parameter JOB_TERMINATE_INTERVAL in `lsb.params`.

- OS-enforced per process limit—When one process in the job exceeds the CPU limit, the limit is enforced by the operating system. For more details, refer to your operating system documentation for `setrlimit()`.

Default Undefined

Notes To make LSB_JOB_CPULIMIT take effect, use the command `badmin hrestart all` to restart all SBDs in the cluster.

Changing the default Terminate job control action—You can define a different terminate action in `lsb.queues` with the parameter JOB_CONTROLS if you do not want the job to be killed. For more details on job controls, see the *LSF Administrator's Guide*.

Limitations If a job is running and the parameter is changed, LSF is not able to reset the type of limit enforcement for running jobs.

- If the parameter is changed from per-process limit enforced by the OS to per-job limit enforced by LSF (LSB_JOB_CPULIMIT=n changed to LSB_JOB_CPULIMIT=y), both per-process limit and per-job limit will affect the running job. This means that signals may be sent to the job either when an individual process exceeds the CPU limit or the sum of the CPU time of all processes of the job exceed the limit. A job that is running may be killed by the OS or by LSF.
- If the parameter is changed from per-job limit enforced by LSF to per-process limit enforced by the OS (LSB_JOB_CPULIMIT=y changed to LSB_JOB_CPULIMIT=n), the job will be allowed to run without limits because the per-process limit was previously disabled.

See Also lsb.queues, bsub, JOB_TERMINATE_INTERVAL, LSB_MOD_ALL_JOBS

LSB_JOB_MEMLIMIT

Syntax **LSB_JOB_MEMLIMIT = y | n**

Description Determines whether the memory limit is a per-process limit enforced by the OS or whether it is a per-job limit enforced by LSF.

- The per-process limit is enforced by the OS when the memory allocated to one process of the job exceeds the memory limit.
- The per-job limit is enforced by LSF when the sum of the memory allocated to all processes of the job exceeds the memory limit.

This parameter applies to memory limits set when a job is submitted with `bsub -M` *`mem_limit`*, and to memory limits set for queues with MEMLIMIT in `lsb.queues`.

The setting of LSB_JOB_MEMLIMIT has the following effect on how the limit is enforced:

When LSB_JOB_MEMLIMIT is	LSF-enforced per-job limit	OS-enforced per-process limit
y	Enabled	Disabled
n or undefined	Disabled	Enabled

- LSF-enforced per-job limit—When the total memory allocated to all processes in the job exceeds the memory limit, LSF sends the following signals to kill the job: SIGINT, SIGTERM, then SIGKILL. The interval between signals is 10 seconds by default.

  On UNIX, the time interval between SIGINT, SIGKILL, SIGTERM can be configured with the parameter JOB_TERMINATE_INTERVAL in `lsb.params`.

- OS-enforced per process limit—When the memory allocated to one process of the job exceeds the memory limit, the operating system enforces the limit. LSF passes the memory limit to the operating system. Some operating systems apply the memory limit to each process, and some do not enforce the memory limit at all.

  OS memory limit enforcement is only available on systems that support RUSAGE_RSS for `setrlimit()`.

  The following operating systems do not support the memory limit at the OS level and the job will be allowed to run without a memory limit:

- Windows NT
- Sun Solaris 2.x

Default Undefined; per-process memory limit enforced by the OS; per-job memory limit enforced by LSF disabled

Notes To make LSB_JOB_MEMLIMIT take effect, use the command `badmin hrestart all` to restart all SBDs in the cluster.

If LSB_JOB_MEMLIMIT is set, it overrides the setting of the parameter LSB_MEMLIMIT_ENFORCE. The parameter LSB_MEMLIMIT_ENFORCE is ignored.

The difference between LSB_JOB_MEMLIMIT set to y and LSB_MEMLIMIT_ENFORCE set to y is that with LSB_JOB_MEMLIMIT, only the per-job memory limit enforced by LSF is enabled. The per-process memory limit enforced by the OS is disabled. With LSB_MEMLIMIT_ENFORCE set to y, both the per-job memory limit enforced by LSF and the per-process memory limit enforced by the OS are enabled.

Changing the default Terminate job control action—You can define a different Terminate action in `lsb.queues` with the parameter JOB_CONTROLS if you do not want the job to be killed. For more details on job controls, see the *LSF Administrator's Guide*.

Limitations If a job is running and the parameter is changed, LSF is not able to reset the type of limit enforcement for running jobs.

- If the parameter is changed from per-process limit enforced by the OS to per-job limit enforced by LSF (LSB_JOB_MEMLIMIT=n or undefined changed to LSB_JOB_MEMLIMIT=y), both per-process limit and per-job limit will affect the running job. This means that signals may be sent to the job either when the memory allocated to an individual process exceeds the memory limit or the sum of memory allocated to all processes of the job exceed the limit. A job that is running may be killed by LSF.
- If the parameter is changed from per-job limit enforced by LSF to per-process limit enforced by the OS (LSB_JOB_MEMLIMIT=y changed to LSB_JOB_MEMLIMIT=n or undefined), the job will be

allowed to run without limits because the per-process limit was previously disabled.

See Also LSB_MEMLIMIT_ENFORCE, lsb.queues, bsub, JOB_TERMINATE_INTERVAL, LSB_MOD_ALL_JOBS

LSB_LOCALDIR

Syntax **LSB_LOCALDIR =** *dir*

Description Sets the duplicate event logging feature.

It specifies the path to the directory in which the replicated event logging process stores the event files.

LSB_LOCALDIR is used to store the primary copy of batch state information. The contents of LSB_LOCALDIR are copied to a replica in LSB_SHAREDIR which resides on a central file server.

The value of LSB_LOCALDIR is the path to a local directory that exists only on the first master.

For example:

```
LSB_LOCALDIR=/usr/local/lsbatch/eventinfo
```

The replicated event logging feature provides added fault-tolerance to MBD/SBD as LSF will be able to tolerate failures of the file server where LSB_SHAREDIR is located. Two MBD daemons will be started. One of these daemons will use LSB_LOCALDIR to store the primary copy of `lsb.events`, which will then be replicated in the LSB_SHAREDIR directory.

Default Undefined

See Also LSB_SHAREDIR, EVENT_UPDATE_INTERVAL in `lsb.params`

LSB_MAILPROG

Syntax **LSB_MAILPROG =** *file_name*

Description

Path and file name of the mail program used by the Batch system to send email.

This is the electronic mail program that LSF will use to send system messages to the user. In a mixed cluster, you can specify different programs for Windows NT and UNIX.

When LSF needs to send email to users it invokes the program defined by LSB_MAILPROG in `lsf.conf`. You can write your own custom mail program and set LSB_MAILPROG to the path where this program is stored.

You can set this parameter during installation on Windows NT. The LSF administrator can set the parameter as part of cluster reconfiguration.

Provide the name of any mail program. For your convenience, LSF provides the following mail programs:

- `sendmail`: Supports the sendmail protocol on UNIX
- `lsmail.exe`: Supports SMTP and Microsoft Exchange Server protocols on Windows NT

  If `lsmail` is specified, the parameter LSB_MAILSERVER must also be specified.

If this parameter is modified, the LSF administrator must restart SBD on all hosts to retrieve the new value.

- On UNIX:

  LSF Batch normally uses `/usr/lib/sendmail` as the mail transport agent to send mail to users. LSF Batch calls LSB_MAILPROG with two arguments; one argument gives the full name of the sender, and the other argument gives the return address for Batch mail.

  LSB_MAILPROG must read the body of the mail message from the standard input. The end of the message is marked by end-of-file. Any program or shell script that accepts the arguments and input, and delivers the mail correctly, can be used.

  LSB_MAILPROG must be executable by any user.
- On Windows NT:

  If LSB_MAILPROG is not defined, no email is sent.

Examples

```
LSB_MAILPROG = lsmail.exe
```

```
LSB_MAILPROG = /serverA/tools/lsf/bin/unixhost.exe
```

Default `/usr/lib/sendmail` (UNIX)

blank (Windows NT)

See Also LSB_MAILSERVER, LSB_MAILTO

LSB_MAILSERVER

Syntax **LSB_MAILSERVER=***mail_protocol:mail_server*

Description Part of mail configuration on Windows NT.

This parameter only applies when `lsmail` is used as the mail program (LSB_MAILPROG = lsmail.exe).Otherwise, it is ignored.

Both *`mail_protocol`* and *`mail_server`* must be indicated.

Set this parameter to either SMTP or Microsoft Exchange protocol (SMTP or EXCHANGE) and specify the name of the host that is the mail server.

This parameter is set during installation of LSF on Windows NT or is set or modified by the LSF administrator.

If this parameter is modified, the LSF administrator must restart SBD on all hosts to retrieve the new value.

Examples
```
LSB_MAILSERVER = EXCHANGE:Host2@company.com
LSB_MAILSERVER = SMTP:MailHost
```

Default Undefined

See Also LSB_MAILPROG

LSB_MAILSIZE_LIMIT

Syntax **LSB_MAILSIZE_LIMIT =** *email_size_in_KB*

Description Limits the size of the email containing LSF batch job output information.

The LSF Batch system sends job information such as CPU, process and memory usage, job output, and errors in email to the submitting user account. Some batch jobs can create large amounts of output. To prevent large job output files from interfering with your mail system, use LSB_MAILSIZE_LIMIT to set the maximum size in KB of the email containing the job information. Specify a positive integer.

If the size of the job output email exceeds LSB_MAILSIZE_LIMIT, the output is saved to a file under JOB_SPOOL_DIR or to the default job output directory if JOB_SPOOL_DIR is undefined. The email informs users of where the job output is located.

If the `-o` option of `bsub` is used, the size of the job output is not checked against LSB_MAILSIZE_LIMIT.

If you use a custom mail program specified by the LSB_MAILPROG parameter that can use the LSB_MAILSIZE environment variable, it is not necessary to configure LSB_MAILSIZE_LIMIT.

Default By default, LSB_MAILSIZE_LIMIT is not enabled. No limit is set on size of batch job output email.

See Also LSB_MAILPROG, LSB_MAILTO

LSB_MAILTO

Syntax **LSB_MAILTO =** *mail_account*

Description LSF Batch sends electronic mail to users when their jobs complete or have errors, and to the LSF administrator in the case of critical errors in the LSF Batch system. The default is to send mail to the user who submitted the job, on the host on which the daemon is running; this assumes that your electronic mail system forwards messages to a central mailbox.

The LSB_MAILTO parameter changes the mailing address used by LSF Batch. LSB_MAILTO is a format string that is used to build the mailing address.

Common formats are:

- `!U`—Mail is sent to the submitting user's account name on the local host. The substring `!U`, if found, is replaced with the user's account name.
- `!U@company_name.com`—Mail is sent to *`user@company_name`*`.com` on the mail server specified by LSB_MAILSERVER.
- `!U@!H`—Mail is sent to *`user@submission_hostname`*. The substring `!H` is replaced with the name of the submission host.

  This format is valid on UNIX only. It is not supported on Windows NT.

All other characters (including any other '`!`') are copied exactly.

If this parameter is modified, the LSF administrator must restart the SBD daemons on all hosts to retrieve the new value.

Default `!U`

See Also LSB_MAILPROG, LSB_MAILSIZE_LIMIT

LSB_MIG2PEND

Syntax **`LSB_MIG2PEND = 0 | 1`**

Description Applies only to migrating jobs.

If 1, requeues migrating jobs intead of restarting or rerunning them on the next available host. Requeues the jobs in the PEND state, in order of the original submission time and job priority, unless LSB_REQUEUE_TO_BOTTOM is also defined.

Does not work with LSF MultiCluster.

Default Undefined

See Also LSB_REQUEUE_TO_BOTTOM

LSB_MBD_PORT

See "LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT" on page 422.

LSB_MEMLIMIT_ENFORCE

Syntax **LSB_MEMLIMIT_ENFORCE = y | n**

Description Specify `y` to enable LSF memory limit enforcement.

If enabled, LSF sends a signal to kill all processes that exceed queue-level memory limits set by MEMLIMIT in `lsb.queues` or job-level memory limits specified by `bsub -M` *mem_limit*.

Otherwise, LSF passes memory limit enforcement to the OS. UNIX operating systems that support RUSAGE_RSS for `setrlimit()` can apply the memory limit to each process. The following operating systems do not support memory limit at the OS level:

- Windows NT
- Sun Solaris 2.x

Default Not defined. LSF passes memory limit enforcement to the OS.

See Also lsb.queues

LSB_MOD_ALL_JOBS

Syntax **LSB_MOD_ALL_JOBS = y | Y**

Description If set, enables `bmod` to modify resource limits and location of job output files for running jobs.

After a job has been dispatched, the following modifications can be made:

- CPU limit (`-c` [*hour*`:`]*minute*[`/`*host_name* | `/`*host_model*] | `-cn`)
- Memory limit (`-M` *mem_limit* | `-Mn`)
- Run limit (`-W` *run_limit*[`/`*host_name* | `/`*host_model*] | `-Wn`)
- Standard output file name (`-o` *output_file* | `-on`)

- Standard error file name (`-e` *error_file* | `-en`)
- Rerunnable jobs (`-r` | `-rn`)

To modify the CPU limit or the memory limit of running jobs, the parameters LSB_JOB_CPULIMIT=Y and LSB_JOB_MEMLIMIT=Y must be defined in `lsf.conf`.

Default Undefined

See Also LSB_JOB_CPULIMIT, LSB_JOB_MEMLIMIT

LSB_NCPU_ENFORCE

Description When set to 1, enables parallel fairshare (considers the number of CPUs when calculating dynamic priority).

Default Undefined

LSB_NQS_PORT

Syntax **`LSB_NQS_PORT =`** *port_number*

Description Required for LSF to work with NQS.

TCP service port to use for communication with NQS.

Where Defined This parameter can alternatively be set as an environment variable or in the services database such as `/etc/services`.

Example
```
LSB_NQS_PORT=607
```

Default Undefined

LSB_QUERY_PORT

Syntax **LSB_QUERY_PORT =** *port_number*

Description Optional. Applies only to UNIX platforms that support thread programming.

This parameter is recommended for busy clusters with many jobs and frequent query requests to increase MBD performance when the following command is used:

- `bjobs`

This may indirectly increase overall MBD performance.

The *`port_number`* is the TCP/IP port number to be used by MBD to only service query requests from the LSF system. MBD checks the query port during initialization.

If LSB_QUERY_PORT is not defined:

- MBD uses the port specified by LSB_MBD_PORT in `lsf.conf`, or, if LSB_MBD_PORT is not defined, looks into the system services database for port numbers to communicate with other hosts in the cluster.
- For each query request it receives, MBD forks one child MBD to service the request. Each child MBD processes one request and then exits.

If LSB_QUERY_PORT is defined:

MBD prepares this port for connection.The default behavior of MBD changes, a child MBD is forked, and the child MBD creates threads to process requests.

MBD responds to requests by forking one child MBD. As soon as MBD has forked a child MBD, the child MBD takes over and listens on the port to process more query requests. For each request, the child MBD creates a thread to process it.

The child MBD continues to listen to the port number specified by LSB_QUERY_PORT and creates threads to service requests until the job changes status, a new job is submitted, or the time specified in

MBD_REFRESH_TIME in `lsb.params` has passed (see "MBD_REFRESH_TIME" on page 302 for more details). At this time, the parent MBD sends a message to the child MBD to exit.

The interval used by MBD for forking new child MBDs is specified by the parameter MBD_REFRESH_TIME in `lsb.params`.

Default Undefined

See Also MBD_REFRESH_TIME

LSB_REQUEUE_TO_BOTTOM

Syntax **`LSB_REQUEUE_TO_BOTTOM = 0 | 1`**

Description Optional. If 1, requeues automatically requeued jobs to the bottom of the queue instead of to the top. Also requeues migrating jobs to the bottom of the queue if LSB_MIG2PEND is defined.

Does not work with LSF MultiCluster.

Default Undefined

See Also REQUEUE_EXIT_VALUES, LSB_MIG2PEND

LSB_SBD_PORT

See "LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT" on page 422.

LSB_SHAREDIR

Syntax **`LSB_SHAREDIR =`** *dir*

Description Directory in which LSF Batch and LSF JobScheduler keep job history and accounting logs for each cluster. These files are necessary for correct operation of the system. Like the organization under LSB_CONFDIR, there is one subdirectory for each cluster.

The LSB_SHAREDIR directory must be owned by the LSF administrator. It must be accessible from all hosts that can potentially become the master host, and must allow read and write access from the LSF master host.

The LSB_SHAREDIR directory typically resides on a reliable file server.

Default `LSF_INDEP/work`

See Also LSB_LOCALDIR

LSB_SIGSTOP

Syntax **`LSB_SIGSTOP =`** *signal_name* | *signal_value*

Description Specifies the signal sent by the SUSPEND action in LSF. You can specify a signal name or a number.

If LSB_SIGSTOP is set to anything other than SIGSTOP, the SIGTSTP signal that is normally sent by the SUSPEND action is not sent.

If this parameter is undefined, by default the SUSPEND action in LSF sends the following signals to a job:

- Parallel or interactive jobs—1. SIGTSTP is sent first to allow user programs to catch the signal and clean up. 2. SIGSTOP is sent 10 seconds after SIGTSTP. SIGSTOP cannot be caught by user programs.
- Other jobs—SIGSTOP is sent. SIGSTOP cannot be caught by user programs.

The same set of signals is not supported on all UNIX systems. To display a list of the symbolic names of the signals (without the SIG prefix) supported on your system, use the `kill -l` command.

Example `LSB_SIGSTOP=SIGKILL`

In this example, the SUSPEND action sends the three default signals sent by the TERMINATE action (SIGINT, SIGTERM, and SIGKILL) 10 seconds apart.

Default Undefined. Default SUSPEND action in LSF is sent.

LSB_SHORT_HOSTLIST

Syntax **LSB_SHORT_HOSTLIST = 1**

Description Displays an abbreviated list of hosts in `bjobs` and `bhist` for a parallel job where multiple processes of a job are running on a host. Multiple processes are displayed in the following format:

processes`*hostA`

For example, if a parallel job is running 5 processes on `hostA`, the information is displayed in the following manner:

`5*hostA`

Setting this parameter may improve MBD restart performance and accelerate event replay.

Default Undefined

LSB_TIME_CMD

Syntax **LSB_TIME_CMD =** *timimg_level*

Description The timing level for checking how long batch commands run.

Time usage is logged in milliseconds; specify a positive integer.

Example: `LSB_TIME_CMD=1`

Default Undefined

See Also LSB_TIME_MBD, LSB_TIME_SBD, LSF_TIME_LIM, LSF_TIME_RES

LSB_TIME_MBD

Syntax **LSB_TIME_MBD =** *timing_level*

Description The timing level for checking how long MBD routines run.

Time usage is logged in milliseconds; specify a positive integer.

Example: `LSB_TIME_MBD=1`

Default Undefined

See Also LSB_TIME_CMD, LSB_TIME_SBD, LSF_TIME_LIM, LSF_TIME_RES

LSB_TIME_SBD

Syntax **`LSB_TIME_SBD =`** *timing_level*

Description The timing level for checking how long SBD routines run.

Time usage is logged in milliseconds; specify a positive integer.

Example: `LSB_TIME_SBD=1`

Default Undefined

See Also LSB_TIME_CMD, LSB_TIME_MBD, LSF_TIME_LIM, LSF_TIME_RES

LSB_UTMP

Syntax **`LSB_UTMP = y | Y`**

Description If set, enables registration of user and account information for interactive batch jobs submitted with `bsub -Ip` or `bsub -Is`. To disable `utmp` file registration, set LSB_UTMP to any value other than `y` or `Y`; for example, `LSB_UTMP=N`.

LSF registers interactive batch jobs the job by adding a entries to the `utmp` file on the execution host when the job starts. After the job finishes, LSF removes the entries for the job from the `utmp` file.

Default Undefined

LSF_AFS_CELLNAME

Syntax **LSF_AFS_CELLNAME =** *AFS_cell_name*

Description Must be defined to AFS cell name if the AFS file system is in use.

Example:

```
LSF_AFS_CELLNAME=cern.ch
```

Default Undefined

LSF_ALARMDIR

Syntax **LSF_ALARMDIR =** *path*

Description JobScheduler only. Specifies the path to the alarm daemon binary.

Default LSF_SERVERDIR, where the default for LSF_SERVERDIR is `/usr/local/lsf/etc.`

See Also LSF_SERVERDIR

LSF_AM_OPTIONS

Syntax **LSF_AM_OPTIONS = AMFIRST | AMNEVER**

Description Determines the order of file path resolution when setting the user's home directory.

This variable is rarely used but sometimes LSF does not properly change the directory to the user's home directory when the user's home directory is automounted. Setting LSF_AM_OPTIONS forces the Batch system to change directory to $HOME before attempting to automount the user's home.

When this parameter is undefined or set to AMFIRST, LSF:

- Sets the user's $HOME directory from the automount path. If it cannot do so, LSF sets the user's $HOME directory from the `passwd` file.

When this parameter is set to AMNEVER, LSF:

- Never uses automount to set the path to the user's home. LSF sets the user's $HOME directory directly from the `passwd` file.

Valid Values The two values are AMFIRST and AMNEVER

Default Undefined; same as AMFIRST

LSF_API_CONNTIMEOUT

Syntax **LSF_API_CONNTIMEOUT =** *seconds*

Description Timeout when connecting to LIM.

Default 5

See Also LSF_API_RECVTIMEOUT

LSF_API_RECVTIMEOUT

Syntax **LSF_API_RECVTIMEOUT =** *seconds*

Description Timeout when receiving a reply from LIM.

Default 20

See Also LSF_API_CONNTIMEOUT

LSF_AUTH

Syntax **LSF_AUTH = eauth | setuid | ident**

Description Optional. Determines the type of authentication used by LSF.

By default, external user authentication is used, and LSF_AUTH is defined to be `eauth`.

External authentication is the only way to provide security for clusters that contain Windows NT hosts.

If this parameter is changed, all LSF daemons must be shut down and restarted by running `lsf_daemons start` on each of the LSF server hosts so that the daemons will use the new authentication method.

If LSF_AUTH is not defined, RES will only accept requests from privileged ports. When LSF uses privileged ports for user authentication, LSF commands must be installed `setuid` to `root` to operate correctly. If the LSF commands are installed in an NFS mounted shared file system, the file system must be mounted with setuid execution allowed (that is, without the `nosuid` option). See the man page for `mount` for more details.

Windows NT does not have the concept of `setuid` binaries and does not restrict access to privileged ports, so the undefined method does not provide any security on Windows NT.

Valid Values

- `eauth`
  For site-specific external authentication.
- `setuid`
  For privileged ports (`setuid`) authentication. This is the mechanism most UNIX remote utilities use. The LSF commands must be installed as `setuid` programs and owned by `root`.
- `ident`
  For authentication using the RFC 931/1413/1414 protocol to verify the identity of the remote client.

If LSF_AUTH is defined as `ident`, RES uses the RFC 1413 identification protocol to verify the identity of the remote user. RES is also compatible with the older RFC 931 authentication protocol. The name, `ident`, must be registered in the system services database.

Default `eauth`

LSF_AUTH_DAEMONS

Syntax **`LSF_AUTH_DAEMONS =`** *any_value*

Description Enables daemon authentication, as long as LSF_AUTH in `lsf.conf` is set to `eauth`. Daemons will call `eauth` to authenticate each other.

Default Undefined

LSF_BINDIR

Syntax **`LSF_BINDIR =`** *dir*

Description Directory in which all LSF user commands are installed.

Default `LSF_MACHDEP/bin`

LSF_CMD_LOGDIR

Syntax **`LSF_CMD_LOGDIR =`** *path*

Description The path to the log files used for debugging LSF commands.

This parameter can also be set from the command line.

Default `/tmp`

See Also LSB_DEBUG_CMD, LSB_CMD_LOGDIR

LSF_CONF_RETRY_INT

Syntax **LSF_CONF_RETRY_INT =** *seconds*

Description The number of seconds to wait between unsuccessful attempts at opening a configuration file (only valid for LIM). This allows LIM to tolerate temporary access failures.

Default 30

See Also LSF_CONF_RETRY_MAX

LSF_CONF_RETRY_MAX

Syntax **LSF_CONF_RETRY_MAX =** *integer*

Description The maximum number of unsuccessful attempts at opening a configuration file (only valid for LIM). This allows LIM to tolerate temporary access failures.

Default 0

See Also LSF_CONF_RETRY_INT

LSF_CONFDIR

Syntax **LSF_CONFDIR =** *dir*

Description Directory in which all LSF configuration files are installed. These files are shared throughout the system and should be readable from any host. This directory can contain configuration files for more than one cluster.

The files in the LSF_CONFDIR directory must be owned by the primary LSF administrator, and readable by all LSF server hosts.

Default `LSF_INDEP/conf`

See Also LSB_CONFDIR

LSF_CROSS_UNIX_NT

Obsolete. Replaced in version 4.1 by LSF_USER_DOMAIN in `lsf.conf`.

LSF_DAEMON_WRAP

Syntax **`LSF_DAEMON_WRAP = y | Y`**

Description Applies only to DCE/DFS and AFS environments; if you are installing LSF on a DCE or AFS environment, set this parameter to `y` or `Y`.

When this parameter is set to `y` or `Y`, MBD, SBD, and RES run the executable `daemons.wrap` in $LSF_SERVERDIR.

Default Undefined

LSF_DEBUG_LIM

Syntax **`LSF_DEBUG_LIM =`** *log_class*

Description Sets the log class for debugging LIM.

Specifies the log class filtering that will be applied to LIM. Only messages belonging to the specified log class are recorded.

The LSF_DEBUG_LIM (which sets the log class) is used in combination with LSF_LOG_MASK (which sets the log level). For example:

```
LSF_LOG_MASK=LOG_DEBUG
LSF_DEBUG_LIM=LC_TRACE
```

You need to restart the daemons after setting LSF_DEBUG_LIM for your changes to take effect.

If you use the command `lsadmin limdebug` to temporarily change this parameter without changing `lsf.conf`, you will not need to restart the daemons.

The daemons log to the `syslog` facility unless LSF_LOGDIR is defined.

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSF_DEBUG_LIM="LC_TRACE LC_EXEC"
```

This parameter can also be defined from the command line.

Valid Values Valid log classes are:

- LC_AFS - Log AFS messages
- LC_AUTH - Log authentication messages
- LC_CHKPNT - log checkpointing messages
- LC_COMM - Log communication messages
- LC_DCE - Log messages pertaining to DCE support
- LC_EXEC - Log significant steps for job execution
- LC_FILE - Log file transfer messages
- LC_HANG - Mark where a program might hang
- LC_LICENCE - Log licence management messages
- LC_MULTI - Log messages pertaining to MultiCluster
- LC_PIM - Log PIM messages
- LC_SCHED - Log JobScheduler messages
- LC_SIGNAL - Log messages pertaining to signals
- LC_TRACE - Log significant program walk steps
- LC_XDR - Log everything transferred by XDR

Default Undefined

See Also LSF_DEBUG_RES, LSF_LOG_MASK, LSF_LOGDIR

LSF_DEBUG_RES

Syntax **LSF_DEBUG_RES =** *log_class*

Description Sets the log class for debugging RES.

Specifies the log class filtering that will be applied to RES. Only messages belonging to the specified log class are recorded.

LSF_DEBUG_RES (which sets the log class) is used in combination with LSF_LOG_MASK (which sets the log level). For example:

```
LSF_LOG_MASK=LOG_DEBUG
LSF_DEBUG_RES=LC_TRACE
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSF_DEBUG_RES="LC_TRACE LC_EXEC"
```

You need to restart the daemons after setting LSF_DEBUG_RES for your changes to take effect.

If you use the command `lsadmin resdebug` to temporarily change this parameter without changing `lsf.conf`, you will not need to restart the daemons.

The daemons log to the `syslog` facility unless LSF_LOGDIR is defined.

This parameter can also be defined from the command line.

Valid Values For a list of valid log classes see LSF_DEBUG_LIM

Default Undefined

See Also LSF_DEBUG_LIM, LSF_LOG_MASK, LSF_LOGDIR

LSF_DEFAULT_EXTSCHED

Syntax **LSF_DEFAULT_EXTSCHED = "CPU_LIST=*cpu_ID_list*; CPUSET_OPTIONS=*option_list*"**

where

- *cpu_ID_list* is a list of CPU IDs separated by commas. The CPU ID is a positive integer or a range of integers. If incorrect CPU IDs are specified, the job is rejected.
- *option_list* is a list of cpuset attributes separated by commas. If incorrect cpuset attributes are specified, the job is rejected.

Description Used with SGI IRIX cpuset support. If set, and the job is submitted without `-extsched` options, the options specified in LSF_DEFAULT_EXTSCHED are used to allocate the cpuset.

LSF creates an IRIX cpuset with the specified CPUs and cpuset attributes and attaches it to the processes of the job when job is scheduled and dispatched.

Default Undefined

LSF_DEFAULT_INSTALL

Syntax **LSF_DEFAULT_INSTALL = y|n**

Description This parameter is set to `y` if the default installation is used; set to `n` otherwise.

Valid Values `y | n`

LSF_DHCP_ENV

Syntax **LSF_DHCP_ENV = y**

Description If defined, enables dynamic IP addressing for all LSF client hosts in the cluster.

Default Undefined

LSF_ENABLE_CSA

Syntax **LSF_ENABLE_CSA = y | Y**

Description If set, enables LSF to write records for LSF jobs to IRIX 6.5.9 Comprehensive System Accounting facility (CSA).

The IRIX 6.5.9 Comprehensive System Accounting facility (CSA) writes an accounting record for each process in the `pacct` file, which is usually located in the `/var/adm/acct/day` directory. IRIX system administrators then use the `csabuild` command to organize and present the records on a job by job basis.

When LSF_ENABLE_CSA is set, for each job run on the IRIX system, LSF writes an LSF-specific accounting record to CSA when the job starts, and when the job finishes. LSF daemon accounting in CSA starts and stops with the LSF daemon.

To disable IRIX CSA accounting, remove LSF_ENABLE_CSA from `lsf.conf`.

See the IRIX 6.5.9 resource administration documentation for information about CSA.

Default Undefined

LSF_ENABLE_EXTSCHEDULER

Syntax **`LSF_ENABLE_EXTSCHEDULER = y | Y`**

Description If set, enables MBD external scheduling for IRIX cpusets.

Default Undefined

LSF_ENVDIR

Syntax **`LSF_ENVDIR =`** *dir*

Description Directory containing the `lsf.conf` file.

By default, `lsf.conf` is installed by creating a shared copy in LSF_CONFDIR and adding a symbolic link from `/etc/lsf.conf` to the shared copy. If LSF_ENVDIR is set, the symbolic link is installed in `LSF_ENVDIR/lsf.conf`.

The `lsf.conf` file is a global environment configuration file for all LSF services and applications. The LSF default installation places the file in LSF_CONFDIR.

Default `/etc`

LSF_EVENT_PROGRAM

Syntax **LSF_EVENT_PROGRAM =** *event_program_name*

Description Specifies the name of the LSF event program to use.

If a full path name is not provided, the default location of this program is LSF_SERVERDIR.

If a program that does not exist is specified, event generation will not work.

If this parameter is undefined, the default name is `genevent` on UNIX and `genevent.exe` on Windows NT.

Default Undefined

LSF_EVENT_RECEIVER

Syntax **LSF_EVENT_RECEIVER =** *event_receiver_program_name*

Description Specifies the LSF event receiver and enables event generation.

Any string may be used as the LSF event receiver; this information is not used by LSF to enable the feature but is only passed as an argument to the event program.

If LSF_EVENT_PROGRAM specifies a program that does not exist, event generation will not work.

If this parameter is undefined, event generation is disabled.

Default Undefined

LSF_ID_PORT

Syntax **LSF_ID_PORT =** *port_number*

Description The network port number used to communicate with the authentication daemon when LSF_AUTH is set to `ident`.

LSF_INCLUDEDIR

Syntax **LSF_INCLUDEDIR =** *dir*

Description Directory under which the LSF API header files `lsf.h` and `lsbatch.h` are installed.

Default `LSF_INDEP/include`

See Also LSF_INDEP

LSF_INDEP

Syntax **LSF_INDEP =** *dir*

Description Specifies the default top-level directory for all machine-independent LSF files.

This includes man pages, configuration files, working directories, and examples. For example, defining LSF_INDEP as `/usr/local/lsf/mnt` places man pages in `/usr/local/lsf/mnt/man`, configuration files in `/usr/local/lsf/mnt/conf`, and so on.

The files in LSF_INDEP can be shared by all machines in the cluster.

As shown in the following list, LSF_INDEP is incorporated into other LSF environment variables.

- LSB_SHAREDIR=$LSF_INDEP/work
- LSF_CONFDIR=$LSF_INDEP/conf
- LSF_INCLUDEDIR=$LSF_INDEP/include
- LSF_MANDIR=$LSF_INDEP/man
- XLSF_APPDIR=$LSF_INDEP/misc

Default `/usr/local/lsf/mnt`

See Also LSF_MACHDEP, LSB_SHAREDIR, LSF_CONFDIR, LSF_INCLUDEDIR, LSF_MANDIR, XLSF_APPDIR

LSF_INTERACTIVE_STDERR

Syntax **LSF_INTERACTIVE_STDERR = y | n**

Description Separates `stderr` from `stdout` for interactive tasks and interactive batch jobs.

This is useful to redirect output to a file with regular UNIX or Windows NT operators instead of the `bsub -e` *`err_file`* and `-o` *`out_file`* options.

This parameter can also be enabled or disabled as an environment variable.

Warning **If you enable this parameter globally in lsf.conf, check any custom scripts that manipulate `stderr` and `stdout`.**

When this parameter is undefined or set to `n`, the following are written to `stdout` on the submission host for interactive tasks and interactive batch jobs:

- Job standard output messages
- Job standard error messages

The following are written to `stderr` on the submission host for interactive tasks and interactive batch jobs:

- LSF messages
- NIOS standard messages
- NIOS debug messages (if LSF_NIOS_DEBUG=1 in `lsf.conf`)

When this parameter is set to `y`, the following are written to `stdout` on the submission host for interactive tasks and interactive batch jobs:

- Job standard output messages

The following are written to `stderr` on the submission host:

- Job standard error messages
- LSF messages
- NIOS standard messages
- NIOS debug messages (if LSF_NIOS_DEBUG=1 in `lsf.conf`)

Default Undefined

Notes When this parameter is set, the change affects interactive tasks and interactive batch jobs run with the following commands:

- `bsub -I`
- `bsub -Ip`
- `bsub -Is`
- `lsrun`
- `lsgrun`
- `lsmake` (LSF Make)
- `bsub pam` (LSF Parallel)

Limitations

- Pseudo-terminal—Do not use this parameter if your application depends on `stderr` as a terminal. This is because LSF must use a non-pseudo-terminal connection to separate `stderr` from `stdout`.
- Synchronization—Do not use this parameter if you depend on messages in `stderr` and `stdout` to be synchronized and jobs in your environment are continuously submitted. A continuous stream of messages causes `stderr` and `stdout` to not be synchronized. This can be emphasized with parallel jobs. This situation is similar to that of `rsh`.
- NIOS standard and debug messages—NIOS standard messages, and debug messages (when LSF_NIOS_DEBUG=1 in `lsf.conf` or as an environment variable) are written to `stderr`. NIOS standard messages are in the format `<<message>>`, which makes it easier to remove them if you wish. To redirect NIOS debug messages to a file, define LSF_CMD_LOGDIR in `lsf.conf` or as an environment variable.

See Also LSF_NIOS_DEBUG, LSF_CMD_LOGDIR

LSF_IRIX_BESTCPUS

Syntax **`LSF_IRIX_BESTCPUS = y | Y`**

Description If set, enables the best-fit algorithm for IRIX cpusets

Default Undefined

LSF_LIBDIR

Syntax **LSF_LIBDIR =** *dir*

Description Specifies the directory in which the LSF libraries are installed. Library files are shared by all hosts of the same type.

Default `LSF_MACHDEP/lib`

LSF_LICENSE_FILE

Syntax **LSF_LICENSE_FILE =** *file_name...* | *port_number@host_name*

Description Specifies either the location of one or more FLEXlm license files used by LSF.

The value for LSF_LICENSE_FILE can be either of the following:

- The full path name to the license file.

  For example, on UNIX:

  ```
  LSF_LICENSE_FILE=/usr/local/lsf/cluster1/conf/license.dat
  ```

  On Windows NT:

  ```
  LSF_LICENSE_FILE= C:\licenses\license.dat
  ```

  or

  ```
  LSF_LICENSE_FILE= \\HostA\licenses\license.dat
  ```

- For a permanent license, the name of the license server host and TCP port number used by the `lmgrd` daemon, in the format *port@host_name*. For example:

  ```
  LSF_LICENSE_FILE="1700@hostD"
  ```

  The port number must be the same as that specified in the `SERVER` line of the license file.

Multiple license files should be quoted and must be separated by a pipe character (|): for example:

```
LSF_LICENSE_FILE= "C:\licenses\license1|C:\licenses\license2|D:\mydir\license3"
```

Multiple files may be kept in the same directory, but each one must reference a different license server. When checking out a license, LSF searches the servers in the order in which they are listed, so it checks the second server when there are no more licenses available from the first server.

If this parameter is not defined, LSF assumes the default location.

Default If you installed LSF with a default installation, the license file is installed in the LSF configuration directory (`LSF_CONFDIR/license.dat`).

If you installed LSF with a custom installation, you specify the license installation directory. The default is the LSF configuration directory (LSF_SERVERDIR for the custom installation).

If you installed FLEXlm separately from LSF to manage other software licenses, the default FLEXlm installation puts the license file in the following locations:

- UNIX: `/usr/local/flexlm/licenses/license.dat`
- Windows NT: `C:\flexlm\licensed`

LSF_LIM_DEBUG

Syntax **`LSF_LIM_DEBUG = 1 | 2`**

Description Sets LSF to debug mode.

If LSF_LIM_DEBUG is defined, LIM operates in single user mode. No security checking is performed, so LIM should not run as `root`.

LIM will not look in the services database for the LIM service port number. Instead, it uses port number 36000 unless LSF_LIM_PORT has been defined.

Specify 1 for this parameter unless you are testing LSF.

Valid Values

- LSF_LIM_DEBUG=1
  LIM runs in the background with no associated control terminal.
- LSF_LIM_DEBUG=2
  LIM runs in the foreground and prints error messages to `tty`.

Default Undefined

See Also LSF_RES_DEBUG

LSF_LIM_PLUGINDIR

Syntax **LSF_LIM_PLUGINDIR =** *path*

Description The path to `liblimvcl.so`. Used only with SUN HPC.

Default $LSF_LIBDIR

See Also LSF_RES_PLUGINDIR

LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT

Syntax Example: **LSF_LIM_PORT =** *port_number*

Description TCP service ports to use for communication with the LSF daemons.

If port parameters are undefined, LSF obtains the port numbers by looking up the LSF service names in the `/etc/services` file or the NIS (UNIX). If it is not possible to modify the services database, you can define these port parameters to set the port numbers.

With careful use of these settings along with the LSF_ENVDIR and PATH environment variables, it is possible to run two versions of the LSF software on a host, selecting between the versions by setting the PATH environment variable to include the correct version of the commands and the LSF_ENVDIR environment variable to point to the directory containing the appropriate `lsf.conf` file.

Default On UNIX, the default is to get port numbers from the services database. On Windows NT, these parameters are mandatory.

Default port number values for both UNIX and Windows NT are:

- LSF_LIM_PORT=6879
- LSF_RES_PORT=6878
- LSB_MBD_PORT=6881

- LSB_SBD_PORT=6882

LSF_LIM_SOL27_PLUGINDIR

Syntax **LSF_LIM_SOL27_PLUGINDIR =** *path*

Description The path to `liblimvcl.so`. Used only with Solaris2.7.

Default $LSF_LIBDIR

See Also LSF_RES_SOL27_PLUGINDIR

LSF_LOG_MASK

Syntax **LSF_LOG_MASK =** *message_log_level*

Description Logging level of messages for LSF daemons.

On UNIX, this is similar to `syslog`. All messages logged at the specified level or higher are recorded; lower level messages are discarded. The LSF_LOG_MASK value can be any log priority symbol that is defined in `syslog.h` (see `syslog`(8)).

The log levels in order from highest to lowest are:

- LOG_EMERG
- LOG_ALERT
- LOG_CRIT
- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

The most important LSF log messages are at the LOG_ERR or LOG_WARNING level. Messages at the LOG_INFO and LOG_DEBUG level are only useful for debugging.

Although message log level implements similar functionalities to UNIX `syslog`, there is no dependency on UNIX `syslog`. It works even if messages are being logged to files instead of `syslog`.

LSF logs error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. The level specified by LSF_LOG_MASK determines which messages are recorded and which are discarded. All messages logged at the specified level or higher are recorded, while lower level messages are discarded.

For debugging purposes, the level LOG_DEBUG contains the fewest number of debugging messages and is used only for very basic debugging. The level LOG_DEBUG3 records all debugging messages, and is not often used. Most debugging is done at the level LOG_DEBUG2.

In versions prior to LSF 4.0, you needed to restart the daemons after setting LSF_LOG_MASK in order for your changes to take effect.

LSF 4.0 implements dynamic debugging, which means you do not need to restart the daemons after setting a debugging environment variable.

The daemons log to the `syslog` facility unless LSF_LOGDIR is defined.

Default LOG_WARNING

See Also LSF_DEBUG_LIM, LSF_DEBUG_RES, LSB_DEBUG_MBD, LSB_DEBUG_SBD, LSB_DEBUG_NQS, LSB_DEBUG_CMD, LSB_DEBUG_CMD, LSF_CMD_LOGDIR

LSF_LOGDIR

Syntax **LSF_LOGDIR =** *dir*

Description This is an optional definition on UNIX and a mandatory parameter on Windows NT.

Error messages from all servers are logged into files in this directory. If a server is unable to write in this directory, the error logs are created in `/tmp` on UNIX.

On Windows NT, if a server is unable to write in this directory, LSF attempts to write in the following directories, in this order:

- LSF_TMPDIR if defined
- `%TMP%` if defined
- `%TEMP%` if defined
- System directory, such as `c:\winnt` for example

(UNIX only) If LSF_LOGDIR is not defined, then `syslog` is used to log everything to the system log using the LOG_DAEMON facility. The syslog facility is available by default on most UNIX systems. The `/etc/syslog.conf` file controls the way messages are logged and the files they are logged to. See the man pages for the `syslogd` daemon and the `syslog` function for more information.

To effectively use debugging, set LSF_LOGDIR to a directory such as `/tmp`. This can be done in your own environment from the shell or in `lsf.conf`.

Default On UNIX, if undefined, log messages go to `syslog`.

On Windows NT, if undefined, no logging is performed.

See Also LSF_LOG_MASK

Files

- `alarmd.log.`*`host_name`* (JobScheduler only)
- `lim.log.`*`host_name`*
- `res.log.`*`host_name`*
- `sbatchd.log.`*`host_name`*
- `sbatchdc.log.`*`host_name`* (Windows NT only)
- `mbatchd.log.`*`host_name`*
- `eeventd.log.`*`host_name`*
- `pim.log.`*`host_name`*

LSF_MACHDEP

Syntax **LSF_MACHDEP =** *dir*

Description Specifies the directory in which machine-dependent files are installed. These files cannot be shared across different types of machines.

In clusters with a single host type, LSF_MACHDEP is usually the same as LSF_INDEP. The machine dependent files are the user commands, daemons, and libraries. You should not need to modify this parameter.

As shown in the following list, LSF_MACHDEP is incorporated into other LSF variables.

- LSF_BINDIR=$LSF_MACHDEP/bin
- LSF_LIBDIR=$LSF_MACHDEP/lib
- LSF_SERVERDIR=$LSF_MACHDEP/etc
- XLSF_UIDDIR=$LSF_MACHDEP/lib/uid

Default `/usr/local/lsf`

See Also LSF_INDEP

LSF_MANDIR

Syntax **LSF_MANDIR =** *dir*

Description Directory under which all man pages are installed.

The man pages are placed in the `man1`, `man3`, `man5`, and `man8` subdirectories of the LSF_MANDIR directory. This is created by the LSF installation process, and you should not need to modify this parameter.

Man pages are installed in a format suitable for BSD-style `man` commands.

For most versions of UNIX, you should add the directory LSF_MANDIR to your MANPATH environment variable. If your system has a `man` command that does not understand MANPATH, you should either install the man pages in the `/usr/man` directory or get one of the freely available `man` programs.

Default `LSF_INDEP/man`

LSF_MASTER_LIST

Syntax **`LSF_MASTER_LIST = "`***host_name ...*`"`**

Description Optional. Defines a list of hosts that are candidates to become the master host for the cluster.

Listed hosts must be defined in `lsf.cluster.`*`cluster_name`*.

Host names are separated by spaces.

Whenever you reconfigure, only master LIM candidates read `lsf.shared` and `lsf.cluster.`*`cluster_name`* to get updated information. The elected master LIM sends configuration information to slave LIMs.

Default Undefined

LSF_MISC

Syntax **`LSF_MISC =`** *dir*

Description Directory in which miscellaneous machine independent files, such as LSF example source programs and scripts, are installed.

Default `LSF_CONFDIR/misc`

LSF_NIOS_DEBUG

Syntax **`LSF_NIOS_DEBUG = 1`**

Description Turns on NIOS debugging for interactive jobs.

If LSF_NIOS_DEBUG=1, NIOS debug messages are written to standard error.

This parameter can also be defined as an environment variable.

When LSF_NIOS_DEBUG and LSF_CMD_LOGDIR are defined, NIOS debug messages are logged in `nios.log.`*`host_name`*. in the location specified by LSF_CMD_LOGDIR.

If LSF_NIOS_DEBUG is defined, and the directory defined by LSF_CMD_LOGDIR is inaccessible, NIOS debug messages are logged to `/tmp/nios.log.`*`host_name`* instead of `stderr`. On Windows NT, NIOS debug messages are also logged to the temporary directory.

Default Undefined

See Also LSF_CMD_LOGDIR

LSF_NIOS_JOBSTATUS_INTERVAL

Syntax **`LSF_NIOS_JOBSTATUS_INTERVAL=`** *`minutes`*

Description Applies only to interactive batch jobs.

Time interval at which NIOS polls MBD to check if a job is still running. Used to retrieve a job's exit status in the case of an abnormal exit of NIOS, due to a network failure for example.

Use this parameter if you run interactive jobs and you have scripts that depend on an exit code being returned.

When this parameter is undefined and a network connection is lost, MBD cannot communicate with NIOS and the return code of a job is not retrieved.

When this parameter is defined, before exiting, NIOS polls MBD on the interval defined by LSF_NIOS_JOBSTATUS_INTERVAL to check if a job is still running. NIOS continues to poll MBD until it receives an exit code or MBD responds that the job does not exist (if the job has already been cleaned from memory for example).

If an exit code cannot be retrieved, NIOS generates an error message and the code -11.

Valid Values Any integer greater than zero

Default Undefined

Notes Set this parameter to large intervals such as 15 minutes or more so that performance is not negatively affected if interactive jobs are pending for too long. NIOS always calls MBD on the defined interval to confirm that a job is still pending and this may add load to MBD.

Product LSF Batch

See Also Environment variable LSF_NIOS_PEND_TIMEOUT

LSF_NIOS_RES_HEARTBEAT

Syntax **LSF_NIOS_RES_HEARTBEAT=** *minutes*

Description Applies only to interactive non-parallel batch jobs.

Defines how long NIOS waits before sending a message to RES to determine if the connection is still open.

Use this parameter to ensure NIOS exits when a network failure occurs instead of waiting indefinitely for notification that a job has been completed. When a network connection is lost, RES cannot communicate with NIOS and as a result, NIOS does not exit.

When this parameter is defined, if there has been no communication between RES and NIOS for the defined period of time, NIOS sends a message to RES to see if the connection is still open. If the connection is no longer available, NIOS exits.

Valid Values Any integer greater than zero

Default Undefined

Notes The time you set this parameter to depends how long you want to allow NIOS to wait before exiting. Typically, it can be a number of hours or days. Too low a number may add load to the system.

Product LSF Base, LSF Batch

LSF_PAM_HOSTLIST_USE

Syntax **LSF_PAM_HOSTLIST_USE=unique**

Description Used to start applications that use both OpenMP and MPI.

Valid Values `unique`

Default Undefined

Notes You can submit a job to LSF Parallel and LSF will reserve the correct number of processors and PAM will start only 1 process per host. For example, to reserve 32 processors and run on 4 processes per host, resulting in the use of 8 hosts:

```
% bsub -n 16 -R "span[ptile=4]" pam yourOpenMPJob
```

Where Defined This parameter can alternatively be set as an environment variable. For example:

```
setenv LSF_PAM_HOSTLIST_USE unique
```

Product LSF Parallel

LSF_PAM_PLUGINDIR

Syntax **LSF_PAM_PLUGINDIR =** *path*

Description The path to `libpamvcl.so`. Used with SUN HPC and LSF Parallel.

Default $LSF_LIBDIR

See Also LSF_RES_PLUGINDIR

LSF_PAM_USE_ASH

Syntax **LSF_PAM_USE_ASH = y | Y**

Description Enables LSF to use the SGI IRIX Array Session Handles (ASH) to propagate signals to the parallel jobs.

See the IRIX system documentation and the `array_session(5)` man page for more information about array sessions.

Default Undefined

LSF_PIM_INFODIR

Syntax **`LSF_PIM_INFODIR =`** *path*

Description The path to where PIM writes the `pim.info.`*`host_name`* file.

Specifies the path to where the process information is stored. The process information resides in the file `pim.info.`*`host_name`*`.` The PIM also reads this file when it starts up so that it can accumulate the resource usage of dead processes for existing process groups.

Default Undefined. If undefined, the system uses `/tmp`.

LSF_PIM_SLEEPTIME

Syntax **`LSF_PIM_SLEEPTIME =`** *seconds*

Description The reporting period for PIM.

PIM updates the process information every 15 minutes unless an application queries this information. If an application requests the information, PIM will update the process information every LSF_PIM_SLEEPTIME seconds. If the information is not queried by any application for more than 5 minutes, the PIM will revert back to the 15 minute update period.

Default 15

LSF_RES_ACCT

Syntax **LSF_RES_ACCT =** *milliseconds* | **0**

Description If this parameter is defined, RES will log information for completed and failed tasks by default (see `lsf.acct`(5)).

The value for LSF_RES_ACCT is specified in terms of consumed CPU time (milliseconds). Only tasks that have consumed more than the specified CPU time will be logged.

If this parameter is defined as LSF_RES_ACCT=0, then all tasks will be logged.

For those tasks that consume the specified amount of CPU time, RES generates a record and appends the record to the task log file `lsf.acct.`*`host_name`*. This file is located in the LSF_RES_ACCTDIR directory.

If this parameter is not defined, the LSF administrator must use the `lsadmin` command (see `lsadmin`(8)) to turn task logging on after RES has started up.

Default Undefined

See Also LSF_RES_ACCTDIR

LSF_RES_ACCTDIR

Syntax **LSF_RES_ACCTDIR =** *dir*

Description The directory in which the RES task log file `lsf.acct.`*`host_name`* is stored.

If LSF_RES_ACCTDIR is not defined, the log file is stored in the `/tmp` directory.

Default (UNIX)`/tmp`

(Windows NT) `C:\temp`

See Also LSF_RES_ACCT

LSF_RES_DEBUG

Syntax **LSF_RES_DEBUG = 1 | 2**

Description Sets RES to debug mode.

If LSF_RES_DEBUG is defined, the Remote Execution Server (RES) will operate in single user mode. No security checking is performed, so RES should not run as `root`. RES will not look in the services database for the RES service port number. Instead, it uses port number 36002 unless LSF_RES_PORT has been defined.

Specify 1 for this parameter unless you are testing RES.

Valid Values
- LSF_RES_DEBUG=1
  RES runs in the background with no associated control terminal.
- LSF_RES_DEBUG=2
  RES runs in the foreground and prints error messages to `tty`.

Default Undefined

See Also LSF_LIM_DEBUG

LSF_RES_PLUGINDIR

Syntax **LSF_RES_PLUGINDIR =** *path*

Description The path to `lsbresvcl.so`. Used only with SUN HPC.

Default $LSF_LIBDIR

See Also LSF_PAM_PLUGINDIR, LSF_LIM_PLUGINDIR

LSF_RES_PORT

See "LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT" on page 422.

LSF_RES_RLIMIT_UNLIM

Syntax **LSF_RES_RLIMIT_UNLIM = cpu | fsize | data | stack | core | vmem**

Description (LSF Base only) By default, RES sets the hard limits for a remote task to be the same as the hard limits of the local process. This parameter specifies those hard limits which are to be set to unlimited, instead of inheriting those of the local process.

Valid values are `cpu`, `fsize`, `data`, `stack`, `core`, and `vmem`, for cpu, file size, data size, stack, core size, and virtual memory limits, respectively.

Example The following example sets the cpu, core size, and stack hard limits to be unlimited for all remote tasks:

```
LSF_RES_RLIMIT_UNLIM="cpu core stack"
```

Default Undefined

See Also LSF_LIM_SOL27_PLUGINDIR

LSF_RES_SOL27_PLUGINDIR

Syntax **LSF_RES_SOL27_PLUGINDIR =** *path*

Description The path to `libresvcl.so`. Used only used with Solaris2.7.

If you want to link a 64-bit object with RES, then you should set LSF_RES_SOL27_PLUGINDIR.

Default $LSF_LIBDIR

LSF_RES_TIMEOUT

Syntax **LSF_RES_TIMEOUT =** *seconds*

Description Timeout when communicating with RES.

Default 15

LSF_ROOT_REX

Syntax **LSF_ROOT_REX = all | local**

Description UNIX only.

Allows `root` remote execution privileges (subject to identification checking) on remote hosts, for both interactive and batch jobs. Causes RES to accept requests from the superuser (`root`) on remote hosts, subject to identification checking.

- Use the keyword `local` to enable `root` remote execution within the local cluster only.
- Use the keyword `all` only with LSF MultiCluster. It enables `root` access and `root` remote execution within the local cluster and across clusters as long as both clusters have LSF_ROOT_REX set to `all`.

If LSF_ROOT_REX is undefined, remote execution requests from user `root` are refused.

Theory Sites that have separate `root` accounts on different hosts within the cluster should not define LSF_ROOT_REX. Otherwise, this setting should be based on local security policies.

The `lsf.conf` file is host-type specific and not shared across different platforms. You must make sure that `lsf.conf` for all your host types are changed consistently.

To avoid weakening system security, the `root` passwords on the involved machines should be identical when using the `all` setting.

Default Undefined; `root` execution is not allowed.

See Also LSF_TIME_CMD, LSF_AUTH

LSF_SECUREDIR

Syntax **LSF_SECUREDIR =** *path*

Description (Windows NT only; mandatory if using `lsf.sudoers`) Path to the directory that contains the file `lsf.sudoers` (shared on an NTFS file system).

LSF_SERVER_HOSTS

Syntax **LSF_SERVER_HOSTS =** "*host_name ...*"

Description Defines one or more LSF server hosts that the application should contact to find a Load Information Manager (LIM). This is used on client hosts on which no LIM is running on the local host. The LSF server hosts are hosts that run LSF daemons and provide loading-sharing services. Client hosts are hosts that only run LSF commands or applications but do not provide services to any hosts.

If LSF_SERVER_HOSTS is not defined, the application tries to contact the LIM on the local host.

The host names in LSF_SERVER_HOSTS must be enclosed in quotes and separated by white space. For example:

```
LSF_SERVER_HOSTS="hostA hostD hostB"
```

Default Undefined

LSF_SERVERDIR

Syntax **LSF_SERVERDIR =** *dir*

Description Directory in which all server binaries and shell scripts are installed.

These include `lim`, `res`, `nios`, `sbatchd`, `mbatchd`, and, `alarmd` and `eeventd` for LSF JobScheduler only. If you use `elim`, `eauth`, `eexec`, `esub`, etc, they are also installed in this directory.

Default `LSF_MACHDEP/etc`

See Also LSB_ECHKPNT_METHOD_DIR

LSF_SHELL_AT_USERS

Syntax **`LSF_SHELL_AT_USERS = "`***user_name user_name ...*`"`**

Description Applies to `lstcsh` only. Specifies users who are allowed to use `@` for host redirection. Users not specified with this parameter cannot use host redirection in `lstcsh`.

If this parameter is undefined, all users are allowed to use `@` for host redirection in `lstcsh`.

Default Undefined

LSF_STRIP_DOMAIN

Syntax **`LSF_STRIP_DOMAIN =`** *domain_suffix* [`:`*domain_suffix ...*]

Description (Optional) If all of the hosts in your cluster can be reached using short host names, you can configure LSF to use the short host names by specifying the portion of the domain name to remove. If your hosts are in more than one domain or have more than one domain name, you can specify more than one domain suffix to remove, separated by a colon (:).

For example, given this definition of LSF_STRIP_DOMAIN,

```
LSF_STRIP_DOMAIN=.foo.com:.bar.com
```

LSF accepts `hostA`, `hostA.foo.com`, and `hostA.bar.com` as names for host `hostA`, and uses the name `hostA` in all output. The leading period '.' is required.

Example:

```
LSF_STRIP_DOMAIN=.platform.com:.generic.com
```

In the above example, LSF accepts `hostA`, `hostA.platform.com`, and `hostA.generic.com` as names for `hostA`, and uses the name `hostA` in all output.

Setting this parameter only affects host names displayed through LSF, it does not affect DNS host lookup.

Default Undefined

LSF_TIME_CMD

Syntax **LSF_TIME_CMD =** *timimg_level*

Description The timing level for checking how long LSF commands run. Time usage is logged in milliseconds; specify a positive integer.

Default Undefined

See Also LSB_TIME_MBD, LSB_TIME_SBD, LSB_TIME_CMD, LSF_TIME_LIM, LSF_TIME_RES

LSF_TIME_LIM

Syntax **LSF_TIME_LIM =** *timing_level*

Description The timing level for checking how long LIM routines run.

Time usage is logged in milliseconds; specify a positive integer.

Default Undefined

See Also LSB_TIME_CMD, LSB_TIME_MBD, LSB_TIME_SBD, LSF_TIME_RES

LSF_TIME_RES

Syntax **LSF_TIME_RES =** *timing_level*

Description The timing level for checking how long RES routines run.

Time usage is logged in milliseconds; specify a positive integer.

Default Undefined

See Also LSB_TIME_CMD, LSB_TIME_MBD, LSB_TIME_SBD, LSF_TIME_LIM

LSF_TMPDIR

Syntax **LSF_TMPDIR =** *dir*

Description Specifies the path and directory for temporary job output.

When LSF_TMPDIR is defined in `lsf.conf`, LSF creates a temporary directory under the directory specified by LSF_TMPDIR on the execution host when a job is started and sets the temporary directory environment variable for the job.

When LSF_TMPDIR is defined as an environment variable, it overrides the LSF_TMPDIR specified in `lsf.conf`. LSF removes the temporary directory and the files that it contains when the job completes.

The name of the temporary directory has the following format:

```
$LSF_TMPDIR/job_ID.tmpdir
```

On UNIX, the directory has the permission 0700.

After adding LSF_TMPDIR to `lsf.conf`, use `badmin hrestart all` to reconfigure your cluster.

This parameter can also be specified from the command line.

Valid Values Specify any valid UNIX or Windows NT path up to a maximum length of 256 characters. The 256 character maximum path length includes the temporary directories and files that LSF Batch creates as jobs run. The path that you specify for LSF_TMPDIR should be as short as possible to avoid exceeding this limit.

On UNIX, specify an absolute path. For example:

```
LSF_TMPDIR=/usr/share/lsf_tmp
```

On Windows NT, specify a UNC path or a path with a drive letter. For example:

```
LSF_TMPDIR=\\HostA\temp\lsf_tmpor
```

```
LSF_TMPDIR=D:\temp\lsf_tmp
```

Default By default, LSF_TMPDIR is not enabled. If LSF_TMPDIR is not specified either in the environment or in `lsf.conf`, this parameter is defined as follows:

- On UNIX: $TMPDIR or `/tmp`
- On Windows NT: %TMP%, %TEMP, or %SystemRoot%

LSF_TOPD_PORT

Syntax **`LSF_TOPD_PORT =`** *port_number*

Description UDP port used for communication between the LSF cpuset topology daemon (`topd`) and the cpuset ELIM. Used with SGI IRIX cpuset support.

Default Undefined

LSF_TOPD_WORKDIR

Syntax **`LSF_TOPD_WORKDIR =`** *directory*

Description Directory to store the IRIX cpuset permission file and the event file for the cpuset topology daemon (`topd`). Used with SGI IRIX cpuset support.

You should avoid using `/tmp` or any other directory that is automatically cleaned up by the system. Unless your installation has restrictions on the LSB_SHAREDIR directory, you should use the default for LSF_TOPD_WORKDIR.

Default `LSB_SHAREDIR/topd_dir.port_number`

Where *port_number* is the value you set for LSF_TOPD_PORT.

LSF_ULDB_DOMAIN

Syntax **LSF_ULDB_DOMAIN =** *domain_name*

Description LSF_ULDB_DOMAIN specifies the name of the LSF domain in the ULDB domain directive. A domain definition of name *domain_name* must be configured in the IRIX `jlimit.in` input file.

Used with IRIX 6.5.8 User Limits Database (ULDB). Configures LSF so that jobs submitted to a host with the IRIX job limits option installed are subject to the job limits configured in the IRIX .

The ULDB contains job limit information that system administrators use to control access to a host on a per user basis. The job limits in the ULDB override the system default values for both job limits and process limits. When a ULDB domain is configured for LSF Batch jobs, LSF limits will be enforced as IRIX job limits.

If the ULDB domain specified in LSF_ULDB_DOMAIN is not valid or does not exist, LSF uses the limits defined in the domain named `batch`. If the `batch` domain does not exist, then the system default limits are set.

See the IRIX 6.5.8 resource administration documentation for information about configuring ULDB domains in the `jlimit.in` file.

Default Undefined

LSF_USE_HOSTEQUIV

Description (UNIX only; optional)

If LSF_USE_HOSTEQUIV is defined, RES and MBD call the `ruserok`(3) function to decide if a user is allowed to run remote jobs.

The `ruserok`(3) function checks in the `/etc/hosts.equiv` file and the user's `$HOME/.rhosts` file to decide if the user has permission to execute remote jobs.

If LSF_USE_HOSTEQUIV is not defined, all normal users in the cluster can execute remote jobs on any host.

If LSF_ROOT_REX is set, `root` can also execute remote jobs with the same permission test as for normal users.

Default Undefined

LSF_USER_DOMAIN

Syntax LSF_USER_DOMAIN = *domain_name* | `.`

Description Set during LSF installation or setup. If you modify this parameter in an existing cluster, you probably have to modify passwords and configuration files also.

Windows or mixed UNIX-Windows clusters only.

Enables default user mapping, and specifies the LSF user domain. The period (`.`) specifies local accounts, not domain accounts.

- a user name specified without a domain is intrepreted (on a Windows host) as belonging to the LSF user domain
- a user name specified with the domain name of the LSF user domain is invalid
- in a mixed cluster, this parameter defines a 2-way, 1:1 user map between UNIX user accounts and Windows user accounts belonging to the specified domain, as long as the accounts have the same user name.

  This means jobs submitted by the Windows user account can run on a UNIX host, and jobs submitted by the UNIX account can run on any Windows host that is available to the Windows user account.

If this parameter is undefined, the default user mapping is not enabled. You can still configure user mapping at the user or system level. User account mapping is required to run cross-platform jobs in a UNIX-Windows mixed cluster.

Default

- If you upgrade from LSF 4.0.1 or earlier, the default is the existing LSF user domain.
- For a new, Windows-only cluster, this parameter is undefined (no LSF user domain, no default user mapping).

- For a new, mixed UNIX-Windows cluster, the default is the domain that the Windows installation account belongs to. This can be modified during LSF installation.

LSF_VPLUGIN

Syntax **LSF_VPLUGIN =** *path*

Description The full path to the SGI vendor MPI library `libxmpi.so`. Used with LSF Parallel and SGI MPI.

For PAM to access the `libxmpi.so` library, the file permission mode must be 755 (`-rwxr-xr-x`).

Default `/usr/lib32/libxmpi.so`

XLSF_APPDIR

Syntax **XLSF_APPDIR =** *dir*

Description (UNIX only; optional) Directory in which X application default files for LSF products are installed.

The LSF commands that use X look in this directory to find the application defaults. Users do not need to set environment variables to use the LSF X applications. The application default files are platform-independent.

Default `LSF_INDEP/misc`

XLSF_UIDDIR

Syntax **XLSF_UIDDIR =** *dir*

Description (UNIX only) Directory in which Motif User Interface Definition files are stored.

These files are platform-specific.

Default `LSF_LIBDIR/uid`

lsf.shared

Overview The `lsf.shared` file contains common definitions that are shared by all load sharing clusters defined by `lsf.cluster.`*`cluster_name`* files. This includes lists of cluster names, host types, host models, the special resources available, and external load indices.

Contents

Cluster Section

(Required) Lists the cluster names recognized by the LSF system

Cluster Section Structure

The first line must contain the mandatory keyword ClusterName. The other keyword is optional.

Each subsequent line defines one cluster.

ClusterName

Description

(Required) Defines all cluster names recognized by the LSF system

All cluster names referenced anywhere in the LSF system must be defined here. The file names of cluster-specific configuration files must end with the associated cluster name.

Servers

Description

(Optional, MultiCluster only) List of hosts in this cluster that LIMs in remote clusters can connect to and obtain information from

For other clusters to work with this cluster, one of these hosts must be running MBD.

By default, the first ten hosts listed in the Host section of `lsf.cluster.`*`cluster_name`* are available to LIMs in remote clusters.

This parameter is useful when LSF_CONFDIR is not shared or replicated.

Cluster Section Example

```
Begin Cluster
ClusterName  Servers
cluster1     hostA
cluster2     hostB
End Cluster
```

HostType Section

(Required) Lists the valid host types in the cluster

HostType Section Structure

The first line consists of the mandatory keyword TYPENAME.

Subsequent lines name valid host types.

TYPENAME

Description Host type names are usually based on a combination of the hardware name and operating system. If your site already has a system for naming host types, you can use the same names for LSF.

HostType Section Example

```
Begin HostType
TYPENAME
SUN41
SOLSPARC
ALPHA
HPPA
NTX86
End HostType
```

HostModel Section

(Required) Lists models of machines and gives the relative CPU scaling factor for each model

LSF uses the relative CPU scaling factor to normalize the CPU load indices so that jobs are more likely to be sent to faster hosts. The CPU factor affects the calculation of job execution time limits and accounting. Using large or inaccurate values for the CPU factor can cause confusing results when CPU time limits or accounting are used.

HostModel Section Structure

The first line consists of the mandatory keywords MODELNAME, CPUFACTOR, and ARCHITECTURE.

Subsequent lines define a model and its CPU factor.

ARCHITECTURE

Description (Reserved for system use only) Indicates automatically detected host models that correspond to the model names.

CPUFACTOR

Description Though it is not required, you would typically assign a CPU factor of 1.0 to the slowest machine model in your system and higher numbers for the others. For example, for a machine model that executes at twice the speed of your slowest model, a factor of 2.0 should be assigned.

MODELNAME

Description Generally, you need to identify the distinct host types in your system, such as MIPS and SPARC first, and then the machine models within each, such as SparcIPC, Sparc1, Sparc2, and Sparc10.

HostModel Section Example

```
Begin HostModel
MODELNAME  CPUFACTOR     ARCHITECTURE
PC400         13.0        (i86pc_400 i686_400)
PC450         13.2        (i86pc_450 i686_450)
Sparc5F        3.0        (SUNWSPARCstation5_170_sparc)
Sparc20        4.7        (SUNWSPARCstation20_151_sparc)
Ultra5S       10.3        (SUNWUltra5_270_sparcv9 SUNWUltra510_270_sparcv9)
End HostModel
```

Resource Section

(Optional) Defines resources.

Resource Section Structure

The first line consists of the keywords. RESOURCENAME and DESCRIPTION are mandatory. The other keywords are optional. Subsequent lines define resources.

RESOURCENAME

Description The name you assign to the new resource. An arbitrary character string.

- A resource name cannot begin with a number.
- A resource name cannot contain any of the following characters:
  `: . ( ) [ + - * / ! & | < > @ =`
- A resource name cannot be any of the following reserved names:
  `cpu cpuf io logins ls idle maxmem maxswp maxtmp type model status it mem ncpus ndisks pg r15m r15s r1m swap swp tmp ut`
- Resource names are case sensitive
- Resource names can be up to 29 characters in length

TYPE

Description The type of resource:

- Boolean—Resources that have a value of 1 on hosts that have the resource and 0 otherwise.
- Numeric—Resources that take numerical values, such as all the load indices, number of processors on a host, or host CPU factor.
- String— Resources that take string values, such as host type, host model, host status.

Default If TYPE is not given, the default type is Boolean.

DESCRIPTION

Description Brief description of the resource.

The information defined here will be returned by the `ls_info`() API call or printed out by the `lsinfo` command as an explanation of the meaning of the resource.

INCREASING

Applies to numeric resources only.

Description If a larger value means greater load, INCREASING should be defined as Y. If a smaller value means greater load, INCREASING should be defined as N.

INTERVAL

Optional. Applies to dynamic resources only.

Description Defines the time interval (in seconds) at which the resource is sampled by the ELIM.

If INTERVAL is defined for a numeric resource, it becomes an external load index.

Default If INTERVAL is not given, the resource is considered static.

RELEASE

Applies to numeric shared resources only, such as floating licenses.

Description Controls whether LSF releases the resource when a job using the resource is suspended. When a job using a shared resource is suspended, the resource is held or released by the job depending on the configuration of this parameter.

Specify N to hold the resource, or specify Y to release the resource.

Default Y

Resource Section Example

```
Begin Resource
RESOURCENAME    TYPE      INTERVAL    INCREASING   RELEASE   DESCRIPTION
mips            Boolean      ()          ()          ()    (MIPS architecture)
dec             Boolean      ()          ()          ()    (DECStation system)
sparc           Boolean      ()          ()          ()    (SUN SPARC)
bsd             Boolean      ()          ()          ()    (BSD unix)
hpux            Boolean      ()          ()          ()    (HP-UX UNIX)
aix             Boolean      ()          ()          ()    (AIX UNIX)
solaris         Boolean      ()          ()          ()    (SUN SOLARIS)
nt              Boolean      ()          ()          ()    (Windows NT)
myResource      String       ()          ()          ()    (MIPS architecture)
static_sh1      Numeric      ()          N           ()    (static)
external_1      Numeric      15          Y           ()    (external)
End Resource
```

lsf.sudoers

Overview

The `lsf.sudoers` file is an optional file to configure security mechanisms for UNIX and Windows.

On Windows, you use `lsf.sudoers` to set the parameter LSF_EAUTH_KEY to configure a key for `eauth` to encrypt and decrypt user authentication data.

On UNIX, you also use `lsf.sudoers` to grant permission to users other than `root` to perform certain operations as `root` in LSF, or as a specified user.

These operations include:

- LSF daemon startup/shutdown
- User ID for LSF authentication
- User ID for LSF pre- and post-execution commands.
- User ID for external LSF executables

If `lsf.sudoers` does not exist, only `root` can perform these operations in LSF on UNIX.

Contents

lsf.sudoers on UNIX

In LSF, certain operations such as daemon startup can only be performed by `root`. The `lsf.sudoers` file grants `root` privileges to specific users or user groups to perform these operations.

Location

`lsf.sudoers` must be located in `/etc` on each host.

Permissions

`lsf.sudoers` must have permission 600 and be readable and writable only by `root`.

lsf.sudoers on Windows

Location

The `lsf.sudoers` file is shared over an NTFS network, not duplicated on every Windows host.

By default, LSF installs `lsf.sudoers` in the %SYSTEMROOT% directory.

The location of `lsf.sudoers` on Windows must be specified by LSF_SECUREDIR in `lsf.conf`. You must configure the LSF_SECUREDIR parameter in `lsf.conf` if using `lsf.sudoers` on Windows.

Permissions

The permissions on `lsf.sudoers` for Windows are:

Workgroup Environment

- Local Admins (W)
- Everyone (R)

Domain Environment

- Domain Admins (W)
- Everyone (R)

File Format

The format of `lsf.sudoers` is very similar to that of `lsf.conf`.

Each entry can have one of the following forms:

- `NAME=VALUE`
- `NAME=`
- `NAME= "STRING1 STRING2 ..."`

The equal sign = must follow each `NAME` even if no value follows and there should be no space beside the equal sign.

`NAME` describes an authorized operation.

`VALUE` is a single string or multiple strings separated by spaces and enclosed in quotation marks.

Lines starting with a pound sign (#) are comments and are ignored.

Example lsf.sudoers File

```
LSB_PRE_POST_EXEC_USER=user100
LSF_STARTUP_PATH=/usr/local/lsf/etc
LSF_STARTUP_USERS="user1 user10 user55"
```

Creating and Modifying lsf.sudoers

You can create and modify `lsf.sudoers` with a text editor such as `vi` or `Notepad`.

On Windows NT, you can use the graphical tool `xlsadmin` to create or modify `lsf.sudoers`, by selecting **Configure** | **Security Parameters**. You must invoke `xlsadmin` as a domain administrator for a Windows NT domain. For a Windows NT workgroup, you must invoke `xlsadmin` as a local user with the necessary administrative privileges.

After you modify `lsf.sudoers`, you need to restart all SBDs in the cluster with the command `badmin hrestart all` to update configuration.

Parameters in lsf.sudoers

LSB_PRE_POST_EXEC_USER

Syntax **LSB_PRE_POST_EXEC_USER =** *user_name*

Description UNIX only.

Specifies the authorized user for running queue level pre-execution and post-execution commands. When this parameter is defined, the queue level pre-execution and post-execution commands will be run as the specified user.

In particular, you can define this parameter if you need to run commands as `root` on UNIX.

Pre- and post-execution commands are configured at the queue level by the LSF administrator.

You can only define a single user name in this parameter.

Default Undefined. Pre- and post-execution commands are run as the user who submitted the job.

LSF_EAUTH_KEY

Syntax **LSF_EAUTH_KEY =** *key*

Description UNIX and Windows NT.

Specifies a key `eauth` uses to encrypt and decrypt user authentication data.

This parameter provides a way to increase security at a site. The rule to choosing a key is the same as for choosing a password.

If you want to improve the security of your site by specifying a key, make sure it is at least six characters long and uses only printable characters (as when choosing a normal UNIX password).

If you want to change the key, modify the `lsf.sudoers` file on every host. For the hosts to work together, they must all use the same key.

Default Undefined. `eauth` encrypts and decrypts authentication data using an internal key.

LSF_EAUTH_USER

Syntax **LSF_EAUTH_USER** = *user_name*

Description UNIX only.

Specifies the user account under which to run the external authentication executable `eauth`.

Default Undefined. `eauth` is run as the primary LSF administrator.

LSF_EEXEC_USER

Syntax **LSF_EEXEC_USER =** *user_name*

Description UNIX only.

Defines the user name to run the external execution command `eexec`.

Default Undefined. `eexec` is run as the user who submitted the job.

LSF_LOAD_PLUGINS

Syntax **LSF_LOAD_PLUGINS=1**

Description Used only for Kerberos authentication in Sun HPC environments. If defined, LSF loads plugins from LSB_LIBDIR.

Ignored by hosts that are not Solaris hosts.

Default Undefined (no plugins).

LSF_STARTUP_USERS

Syntax **LSF_STARTUP_USERS = all_admins** | **"***user_name***..."**

Description UNIX only. Equivalent to the local LSF administrators group (Local Admins) in Windows NT. Must be defined in conjunction with LSF_STARTUP_PATH for this feature to work.

By default, `root` is the only user who can start up the LSF daemons as `root`. `lsadmin` and `badmin` must be installed as `setuid root` programs.

This parameter specifies other users who can start daemons as `root` using the LSF administration commands `lsadmin` and `badmin`.

- **`all_admins`**

  Allows all LSF administrators configured in `lsf.cluster.`*`cluster_name`* to start up LSF daemons as `root` by running `lsadmin` and `badmin` commands.

  Defining LSF_STARTUP_USERS as `all_admins` incurs some security risk because administrators can be configured by a primary LSF administrator who is not `root`. You should explicitly list the login names of all authorized administrators here so that you have full control of who can start daemons as `root`.

- **`"`***`user_name`***`..."`**

  Allows specified users to start up LSF daemons as `root` by running `lsadmin` and `badmin` commands. If only one user is specified, quotation marks are not required.

Default Undefined. Only `root` can start up daemons as `root`.

See Also LSF_STARTUP_PATH

LSF_STARTUP_PATH

Syntax **`LSF_STARTUP_PATH =`** *`path`*

Description UNIX only.

Absolute path name of the directory in which the server binaries (LIM, RES, SBD, MBD, etc.) are installed.

This is normally LSF_SERVERDIR as defined in your `lsf.conf` file. LSF will allow the specified administrators (see "LSF_STARTUP_USERS" on page 459) to start the daemons installed in the LSF_STARTUP_PATH directory.

Both LSF_STARTUP_USERS and LSF_STARTUP_PATH must be defined for this feature to work.

Default Undefined

See Also LSF_STARTUP_USERS

SEE ALSO

`lsadmin`(8), `badmin`(8), `lsf.conf`(5), `lsfstartup`(3), `lsf.cluster`(5), `eexec`(8), `eauth`(8)

lsf.task

Overview

Users should not have to specify a resource requirement each time they submit a job. LSF supports the concept of a task list. This chapter describes the files used to configure task lists:

- `lsf.task`
- `lsf.task.`*`cluster_name`*
- `.lsftask`

Contents

About Task Lists

A task list is a list in LSF that keeps track of the default resource requirements for different applications and task eligibility for remote execution.

The term task refers to an application name. With a task list defined, LSF automatically supplies the resource requirement of the job whenever users submit a job unless one is explicitly specified at job submission.

LSF takes the job's command name as the task name and uses that name to find the matching resource requirement for the job from the task list. If a task does not have an entry in the task list, LSF assumes the default resource requirement; that is, a host that has the same host type as the submission host will be chosen to run the job.

Task Files

LSF task lists can be configured in three files: `lsf.task`(systemwide), `lsf.task.cluster_name`(clusterwide), and `.lsftask`(user-specific).

The clusterwide task file is used to augment the systemwide file. The user's task file is used to augment the systemwide and clusterwide task files.

LSF combines the systemwide, clusterwide, and user-specific task lists for each user's view of the task list. In cases of conflicts, such as different resource requirements specified for the same task in different lists, the clusterwide list overrides the systemwide list, and the user-specific list overrides both.

LSF_CONFDIR/lsf.task

Systemwide task list applies to all clusters and all users.

LSF_CONFDIR/lsf.task.*cluster_name*

Clusterwide task list applies to all users in the same cluster.

$HOME/.lsftask

User task list, one per user, applies only to the specific user. This file is automatically created in the user's home directory whenever a user first updates his task lists using the `lsrtasks` or `lsltasks` commands. For details about task eligibility lists, see the man page `ls_task`(3).

Permissions

Only the LSF administrator can modify the systemwide task list(`lsf.task`) and the clusterwide task list(`lsf.task.`*`cluster_name`*).

A user can modify his own task list(`.lsftask`) with the `lsrtasks` and `lsltasks` commands. See the man pages `lsrtasks`(1) and `lsltasks`(1) for more details.

Format of Task Files

Each file consists of two sections, `LocalTasks` and `RemoteTasks`. For example:

```
Begin LocalTasks
ps
hostname
uname
crontab
End LocalTasks

Begin RemoteTasks
+ "newjob/mem>25"
+ "verilog/select[type==any && swp>100]"
make/cpu
nroff/-
End RemoteTasks
```

Tasks are listed one per line. Each line in a section consists of a task name, and, for the `RemoteTasks` section, an optional resource requirement string separated by a slash (/).

A plus sign (+) or a minus sign (-) can optionally precede each entry. If no `+` or `-` is specified, `+` is assumed.

A `+` before a task name means adding a new entry (if non-existent) or replacing an entry (if already existent) in the task list. A `-` before a task name means removing an entry from the application's task lists if it was already created by reading higher level task files.

LocalTasks Section

The section starts with `Begin LocalTasks` and ends with `End LocalTasks`.

This section lists tasks that are not eligible for remote execution, either because they are trivial tasks or because they need resources on the local host.

RemoteTasks Section

The section starts with `Begin RemoteTasks` and ends with `End RemoteTasks`.

This section lists tasks that are eligible for remote execution. You can associate resource requirements with each task name.

See `lsfintro`(1) for a description of the resource requirement string. If the resource requirement string is not specified for a remote task, the default is `"select[type==local] order[r15s:pg]"`.

SEE ALSO

`lsfintro`(1), `lsrtasks`(1), `lsltasks`(1), `ls_task`(3), `lsf.conf`(5)

PART

IV

Troubleshooting

Troubleshooting and Error Messages

Overview This chapter describes some common problems with LSF and LSF Batch operations, answers some frequently asked questions, and provides some instructions for solving problems.

Contents

Shared File Access

A frequent problem with LSF is non-accessible files due to a non-uniform file space. If a task is run on a remote host where a file it requires cannot be accessed using the same name, an error results. Almost all interactive LSF commands fail if the user's current working directory cannot be found on the remote host.

Shared Files on UNIX

If you are running NFS, rearranging the NFS mount table may solve the problem. If your system is running the `automount` server, LSF tries to map the filenames, and in most cases it succeeds. If shared mounts are used, the mapping may break for those files. In such cases, specific measures need to be taken to get around it.

The automount maps must be managed through NIS. When LSF tries to map filenames, it assumes that automounted file systems are mounted under the `/tmp_mnt` directory.

Shared Files on Windows NT

To share files among Windows NT machines, set up a share on the server and access it from the client. You can access files on the share either by specifying a UNC path (`\\server\share\path`) or connecting the share to a local drive name and using a `drive:\path` syntax. Using UNC is recommended because drive mappings may be different across machines, while UNC allows you to unambiguously refer to a file on the network.

Shared Files Across UNIX and Windows NT

For file sharing across UNIX and Windows NT, you require a third party NFS product on Windows NT to export directories from Windows NT to UNIX.

Common LSF Problems

This section lists some other common problems with the LIM, the RES and interactive applications.

LIM Dies Quietly

Run the following command to check for errors in the LIM configuration files.

```
% lsadmin ckconfig -v
```

This displays most configuration errors. If this does not report any errors, check in the LIM error log.

LIM Unavailable

Sometimes the LIM is up, but executing the `lsload` command prints the following error message:

```
Communication time out.
```

If the LIM has just been started, this is normal, because the LIM needs time to get initialized by reading configuration files and contacting other LIMs.

If the LIM does not become available within one or two minutes, check the LIM error log for the host you are working on.

When the local LIM is running but there is no master LIM in the cluster, LSF applications display the following message:

```
Cannot locate master LIM now, try later.
```

Check the LIM error logs on the first few hosts listed in the `Host` section of the `lsf.cluster.`*`cluster_name`* file. If LSF_MASTER_LIST is defined in `lsf.conf`, check the LIM error logs on the hosts listed in this parameter instead.

RES Does Not Start

Check the RES error log.

UNIX If the RES is unable to read the `lsf.conf` file and does not know where to write error messages, it logs errors into `syslog(3)`.

Windows NT If the RES is unable to read the `lsf.conf` file and does not know where to write error messages, it logs errors into `C:\temp`.

User Permission Denied

If remote execution fails with the following error message, the remote host could not securely determine the user ID of the user requesting remote execution.

```
User permission denied.
```

Check the RES error log on the remote host; this usually contains a more detailed error message.

If you are not using an identification daemon (LSF_AUTH is not defined in the `lsf.conf` file), then all applications that do remote executions must be owned by `root` with the `setuid` bit set. This can be done as follows.

```
% chmod 4755 filename
```

If the binaries are on an NFS-mounted file system, make sure that the file system is not mounted with the `nosuid` flag.

If you are using an identification daemon (defined in the `lsf.conf` file by LSF_AUTH), `inetd` must be configured to run the daemon. The identification daemon must not be run directly.

If LSF_USE_HOSTEQUIV is defined in the `lsf.conf` file, check if `/etc/hosts.equiv` or `HOME/.rhosts` on the destination host has the client host name in it. Inconsistent host names in a name server with `/etc/hosts` and `/etc/hosts.equiv` can also cause this problem.

On SGI hosts running a name server, you can try the following command to tell the host name lookup code to search the `/etc/hosts` file before calling the name server.

```
% setenv HOSTRESORDER "local,nis,bind"
```

Non-uniform File Name Space

A command may fail with the following error message due to a non-uniform file name space.

```
chdir(...) failed: no such file or directory
```

You are trying to execute a command remotely, where either your current working directory does not exist on the remote host, or your current working directory is mapped to a different name on the remote host.

If your current working directory does not exist on a remote host, you should not execute commands remotely on that host.

On UNIX If the directory exists, but is mapped to a different name on the remote host, you have to create symbolic links to make them consistent.

LSF can resolve most, but not all, problems using `automount`. The automount maps must be managed through NIS. Follow the instructions in your Release Notes for obtaining technical support if you are running automount and LSF is not able to locate directories on remote hosts.

Common LSF Batch Problems

This section lists some common problems with LSF Batch. Most problems are due to incorrect installation or configuration. Check the `mbatchd` and `sbatchd` error log files; often the log message points directly to the problem.

Batch Daemons Die Quietly

First, check the `sbatchd` and `mbatchd` error logs. Try running the following command to check the configuration.

```
% badmin ckconfig
```

This reports most errors. You should also check if there is any email from LSF Batch in the LSF administrator's mailbox. If the `mbatchd` is running but the `sbatchd` dies on some hosts, it may be because `mbatchd` has not been configured to use those hosts.

See "Host Not Used By LSF Batch" on page 477.

sbatchd Starts But mbatchd Does Not

Check whether LIM is running. You can test this by running the `lsid` command. If LIM is not running properly, follow the suggestions in this chapter to fix the LIM first. You should make sure that LSF and LSF Batch are using the same `lsf.conf` file. Note that it is possible that `mbatchd` is temporarily unavailable because the master LIM is temporarily unknown, causing the following error message.

```
sbatchd: unknown service
```

Check whether services are registered properly. Refer to the *LSF Administrator's Guide* for information about registering LSF services.

Host Not Used By LSF Batch

If you configure a list of server hosts in the `Host` section of the `lsb.hosts` file, `mbatchd` allows `sbatchd` to run only on the hosts listed. If you try to configure an unknown host in the `HostGroup` or `HostPartition` sections of the `lsb.hosts` file, or as a `HOSTS` definition for a queue in the `lsb.queues` file, `mbatchd` logs the following message.

```
mbatchd on host: LSB_CONFDIR/cluster/configdir/file(line #):
Host hostname is not used by lsbatch;

ignored
```

If you start `sbatchd` on a host that is not known by `mbatchd`, `mbatchd` rejects the `sbatchd`. The `sbatchd` logs the following message and exits.

```
This host is not used by lsbatch system.
```

Both of these errors are most often caused by not running the following commands, in order, after adding a host to the configuration.

```
lsadmin reconfig

badmin reconfig
```

You must run both of these before starting the daemons on the new host.

Error Messages

The following error messages are logged by the LSF daemons, or displayed by the following commands.

```
lsadmin ckconfig
```

```
badmin ckconfig
```

General Errors

The messages listed in this section may be generated by any LSF daemon.

`can't open file: error`

The daemon could not open the named file for the reason given by *error*. This error is usually caused by incorrect file permissions or missing files. All directories in the path to the configuration files must have execute (`x`) permission for the LSF administrator, and the actual files must have read (`r`) permission. Missing files could be caused by incorrect path names in the `lsf.conf` file, running LSF daemons on a host where the configuration files have not been installed, or having a symbolic link pointing to a nonexistent file or directory.

`file(line): malloc failed`

Memory allocation failed. Either the host does not have enough available memory or swap space, or there is an internal error in the daemon. Check the program load and available swap space on the host; if the swap space is full, you must add more swap space or run fewer (or smaller) programs on that host.

`auth_user: getservbyname(ident/tcp) failed: error; ident must be registered in services`

LSF_AUTH=ident is defined in the `lsf.conf` file, but the `ident/tcp` service is not defined in the services database. Add `ident/tcp` to the services database, or remove LSF_AUTH from the `lsf.conf` file and `setuid root` those LSF binaries that require authentication.

`auth_user: operation(<host>/<port>) failed: error`

LSF_AUTH=ident is defined in the `lsf.conf` file, but the LSF daemon failed to contact the `identd` daemon on host. Check that `identd` is defined in `inetd.conf` and the `identd` daemon is running on host.

`auth_user: Authentication data format error (rbuf=<data>) from <host>/<port>`

`auth_user: Authentication port mismatch (...) from <host>/<port>`

LSF_AUTH=ident is defined in the `lsf.conf` file, but there is a protocol error between LSF and the ident daemon on *host*. Make sure the ident daemon on the host is configured correctly.

`userok: Request from bad port (<port_number>), denied`

LSF_AUTH is not defined, and the LSF daemon received a request that originates from a non-privileged port. The request is not serviced.

Set the LSF binaries (for example, `lsrun`) to be owned by `root` with the `setuid` bit set, or define LSF_AUTH=ident and set up an ident server on all hosts in the cluster. If the binaries are on an NFS-mounted file system, make sure that the file system is not mounted with the `nosuid` flag.

`userok: Forged username suspected from <host>/<port>: <claimed_user>/<actual_user>`

The service request claimed to come from user *claimed_user* but ident authentication returned that the user was actually *actual_user*. The request was not serviced.

`userok: ruserok(<host>,<uid>) failed`

LSF_USE_HOSTEQUIV is defined in the `lsf.conf` file, but *host* has not been set up as an equivalent host (see `/etc/host.equiv`), and user *uid* has not set up a `.rhosts` file.

`init_AcceptSock: RES service(res) not registered, exiting`

`init_AcceptSock: res/tcp: unknown service, exiting`

`initSock: LIM service not registered.`

`initSock: Service lim/udp is unknown. Read LSF Guide for help`

`get_ports: <serv> service not registered`

The LSF services are not registered. Refer to the *LSF Administrator's Guide* for information about configuring LSF services.

`init_AcceptSock: Can't bind daemon socket to port <port>: error, exiting`

`init_ServSock: Could not bind socket to port <port>: error`

These error messages can occur if you try to start a second LSF daemon (for example, RES is already running, and you execute RES again). If this is the case, and you want to start the new daemon, kill the running daemon or use the `lsadmin` or `badmin` commands to shut down or restart the daemon.

Configuration Errors

The messages listed in this section are caused by problems in the LSF configuration files. General errors are listed first, and then errors from specific files.

```
file(line): Section name expected after Begin; ignoring section
```

```
file(line): Invalid section name name; ignoring section
```

The keyword `begin` at the specified line is not followed by a section name, or is followed by an unrecognized section name.

```
file(line): section section: Premature EOF
```

The end of file was reached before reading the `end section` line for the named section.

```
file(line): keyword line format error for section section; Ignore this section
```

The first line of the section should contain a list of keywords. This error is printed when the keyword line is incorrect or contains an unrecognized keyword.

```
file(line): values do not match keys for section section; Ignoring line
```

The number of fields on a line in a configuration section does not match the number of keywords. This may be caused by not putting `()` in a column to represent the default value.

```
file: HostModel section missing or invalid
```

```
file: Resource section missing or invalid
```

```
file: HostType section missing or invalid
```

The `HostModel`, `Resource`, or `HostType` section in the `lsf.shared` file is either missing or contains an unrecoverable error.

```
file(line): Name name reserved or previously defined. Ignoring index
```

The name assigned to an external load index must not be the same as any built-in or previously defined resource or load index.

`file(line): Duplicate clustername name in section cluster. Ignoring current line`

A cluster name is defined twice in the same `lsf.shared` file. The second definition is ignored.

`file(line): Bad cpuFactor for host model model. Ignoring line`

The CPU factor declared for the named host model in the `lsf.shared` file is not a valid number.

`file(line): Too many host models, ignoring model name`

You can declare a maximum of 127 host models in the `lsf.shared` file.

`file(line): Resource name name too long in section resource. Should be less than 40 characters. Ignoring line`

The maximum length of a resource name is 39 characters. Choose a shorter name for the resource.

`file(line): Resource name name reserved or previously defined. Ignoring line.`

You have attempted to define a resource name that is reserved by LSF or already defined in the `lsf.shared` file. Choose another name for the resource.

`file(line): illegal character in resource name: name, section resource. Line ignored.`

Resource names must begin with a letter in the set [a-zA-Z], followed by letters, digits or underscores [a-zA-Z0-9_].

LIM Messages

The following messages are logged by the LIM:

`main: LIM cannot run without licenses, exiting`

The LSF software license key is not found or has expired. Check that FLEXlm is set up correctly, or contact your LSF technical support.

`main: Received request from unlicensed host <host>/<port>`

LIM refuses to service requests from hosts that do not have licenses. Either your LSF license has expired, or you have configured LSF on more hosts than your license key allows.

`initLicense: Trying to get license for LIM from source <LSF_CONFDIR/license.dat>`

`getLicense: Can't get software license for LIM from license file <LSF_CONFDIR/license.dat>: feature not yet available.`

Your LSF license is not yet valid. Check whether the system clock is correct.

```
findHostbyAddr/<proc>: Host <host>/<port> is unknown by <myhostname>
```

```
function: Gethostbyaddr_(<host>/<port>) failed: error
```

```
main: Request from unknown host <host>/<port>: error
```

```
function: Received request from non-LSF host <host>/<port>
```

The daemon does not recognize *host* as an LSF host. The request is not serviced. These messages can occur if *host* was added to the configuration files, but not all the daemons have been reconfigured to read the new information. If the problem still occurs after reconfiguring all the daemons, check whether the host is a multi-addressed host. Refer to the *LSF Administrator's Guide* for information about working with multi-addressed hosts.

```
rcvLoadVector: Sender (<host>/<port>) may have different config?
```

```
MasterRegister: Sender (host) may have different config?
```

LIM detected inconsistent configuration information with the sending LIM. Run the following command so that all the LIMs have the same configuration information.

```
% lsadmin reconfig
```

Note any hosts that failed to be contacted.

```
rcvLoadVector: Got load from client-only host <host>/<port>. Kill LIM on
<host>/<port>
```

A LIM is running on an LSF client host. Run the following command, or go to the client host and kill the LIM daemon.

```
% lsadmin limshutdown host
```

```
saveIndx: Unknown index name <name> from ELIM
```

LIM received an external load index name that is not defined in the `lsf.shared` file. If name is defined in `lsf.shared`, reconfigure the LIM. Otherwise, add name to the `lsf.shared` file and reconfigure all the LIMs.

```
saveIndx: ELIM over-riding value of index <name>
```

This is a warning message. The ELIM sent a value for one of the built-in index names. LIM uses the value from ELIM in place of the value obtained from the kernel.

```
getusr: Protocol error numIndx not read (cc=num): error
```

```
getusr: Protocol error on index number (cc=num): error
```

Protocol error between ELIM and LIM. Refer to the *LSF Administrator's Guide* for a description of the ELIM and LIM protocols.

RES Messages

These messages are logged by the RES.

```
doacceptconn: getpwnam(<username>@<host>/<port>) failed: error
```

```
doacceptconn: User <username> has uid <uid1> on client host <host>/<port>, uid
<uid2> on RES host; assume bad user
```

```
authRequest: username/uid <userName>/<uid>@<host>/<port> does not exist
```

```
authRequest: Submitter's name <clname>@<clhost> is different from name <lname> on
this host
```

RES assumes that a user has the same userID and username on all the LSF hosts. These messages occur if this assumption is violated. If the user is allowed to use LSF for interactive remote execution, make sure the user's account has the same userID and username on all LSF hosts.

```
doacceptconn: root remote execution permission denied
```

```
authRequest: root job submission rejected
```

Root tried to execute or submit a job but LSF_ROOT_REX is not defined in the `lsf.conf` file.

```
resControl: operation permission denied, uid = <uid>
```

The user with user ID *uid* is not allowed to make RES control requests. Only the LSF manager, or root if LSF_ROOT_REX is defined in `lsf.conf`, can make RES control requests.

```
resControl: access(respath, X_OK): error
```

The RES received a reboot request, but failed to find the file `respath` to re-execute itself. Make sure `respath` contains the RES binary, and it has execution permission.

LSF Batch Messages

The following messages are logged by the `mbatchd` and `sbatchd` daemons:

```
renewJob: Job <jobId>: rename(<from>,<to>) failed: error
```

`mbatchd` failed in trying to re-submit a rerunnable job. Check that the file *from* exists and that the LSF administrator can rename the file. If *from* is in an AFS directory, check that the LSF administrator's token processing is properly setup

Refer to *LSF Administrator's Guide* for information about installing on AFS.

```
logJobInfo_: fopen(<logdir/info/jobfile>) failed: error

logJobInfo_: write <logdir/info/jobfile> <data> failed: error

logJobInfo_: seek <logdir/info/jobfile> failed: error

logJobInfo_: write <logdir/info/jobfile> xdrpos <pos> failed: error

logJobInfo_: write <logdir/info/jobfile> xdr buf len <len> failed: error

logJobInfo_: close(<logdir/info/jobfile>) failed: error

rmLogJobInfo: Job <jobId>: can't unlink(<logdir/info/jobfile>): error

rmLogJobInfo_: Job <jobId>: can't stat(<logdir/info/jobfile>): error

readLogJobInfo: Job <jobId> can't open(<logdir/info/jobfile>): error

start_job: Job <jobId>: readLogJobInfo failed: error

readLogJobInfo: Job <jobId>: can't read(<logdir/info/jobfile>) size size: error

initLog: mkdir(<logdir/info>) failed: error

<fname>: fopen(<logdir/file> failed: error

getElogLock: Can't open existing lock file <logdir/file>: error

getElogLock: Error in opening lock file <logdir/file>: error

releaseElogLock: unlink(<logdir/lockfile>) failed: error

touchElogLock: Failed to open lock file <logdir/file>: error

touchElogLock: close <logdir/file> failed: error
```

`mbatchd` failed to create, remove, read, or write the log directory or a file in the log directory, for the reason given in *error*. Check that LSF administrator has read, write, and execute permissions on the `logdir` directory.

If `logdir` is on AFS, check that the instructions in the *LSF Administrator's Guide* have been followed. Use the `fs ls` command to verify that the LSF administrator owns `logdir` and that the directory has the correct acl.

```
replay_newjob: File <logfile> at line <line>: Queue <queue> not found, saving to
queue <lost_and_found>
```

```
replay_switchjob: File <logfile> at line <line>: Destination queue <queue> not
found, switching to queue <lost_and_found>
```

When `mbatchd` was reconfigured, jobs were found in *queue* but that queue is no longer in the configuration.

```
replay_startjob: JobId <jobId>: exec host <host> not found, saving to host
<lost_and_found>
```

When mbatchd was reconfigured, the event log contained jobs dispatched to host, but that host is no longer configured to be used by LSF Batch.

```
do_restartReq: Failed to get hData of host <host_name>/<host_addr>
```

`mbatchd` received a request from `sbatchd` on host *host_name*, but that host is not known to `mbatchd`. Either the configuration file has been changed but `mbatchd` has not been reconfigured to pick up the new configuration, or *host_name* is a client host but the `sbatchd` daemon is running on that host. Run the following command to reconfigure the `mbatchd` or kill the `sbatchd` daemon on *host_name*.

```
% badmin reconfig
```

Index

M

N

V

W

X